目　录

第 1 章　系统辨识工具箱

(System Identification Toolbox,Ver 4.0.4)

控制理论经过数十年的发展，取得了丰富的研究成果，并在工程中得到了广泛的应用。目前控制理论的大量研究成果，如经典控制、现代控制和自适应控制等都是基于对控制对象的动力学特性进行一定精度的数学建模，因此尽可能获得控制对象的精确数学模型是成功地进行控制器设计的重要因素。当对象较为简单时，可以应用有关知识建立对象的机理模型，并能够达到较高的精度。随着人类生产的发展，控制对象日益复杂，建立对象的机理模型往往是困难的，许多情况下只能获得对象的输入输出数据。因此，如何利用对象的输入输出数据建立数学模型就成为控制理论和工程界研究的重要内容。有关这方面的研究逐渐形成了一门相对独立的学科——系统辨识学科。

所谓辨识，就是通过测量研究对象在人为输入作用下的输出响应，或正常运行时的输入输出数据记录，加以必要的数据处理和数学计算，估计出对象的数学模型。系统辨识的理论研究已取得大量的成果，其应用领域也超出了控制工程的范畴。

Matlab 的系统辨识工具箱提供了进行系统模型辨识的有力工具，其主要功能包括：

(1) 参数模型辨识工具，包括 AR、ARX、状态空间和输出误差等模型类的辨识工具；

(2) 非参数模型辨识工具；

(3) 模型验证工具，即对辨识模型进行仿真并将真实输出数据与模型预测数据进行比较，计算相应的残差；

(4) 递推参数估计，针对各种参数模型，利用递推估计方法获得模型参数；

(5) 各种模型类的建立和转换函数；

(6) 集成多种功能的图形用户界面，该界面以图形交互方式提供模型类的选择和建立、输入输出数据的加载和预处理，以及模型的估计等功能。

1.1　系统辨识的基本原理和常用辨识模型

1.1.1　系统辨识的基本原理

1　系统辨识的定义和基本要素

1978 年瑞典著名学者 L.Ljung 给出系统辨识的定义：“辨识有三个要素即数据、模型类

和准则，辨识就是按照一个准则在一组模型类中选择一个与数据拟合的最好的模型。”该定义强调了系统辨识的三个基本要素，其中数据是指系统的输入输出数据，模型类则定义了模型的基本结构类型，准则即为评价模型与输入输出数据拟合程度的量度标准。

2 系统辨识的等价准则

等价准则也称为误差准则，是系统辨识问题中的基本要素之一，用来衡量模型接近实际过程的标准，通常被表示为辨识模型与实际对象模型的误差的泛函。这里所说的误差可以是输出误差、输入误差或广义误差。

3 辨识的内容和步骤

系统辨识的内容主要包括以下四个方面：

（1）实验设计；

（2）模型结构辨识；

（3）模型参数辨识；

（4）模型检验。

• 实验设计

系统辨识实验设计需要完成的工作包括选择和确定输入信号、采样时间、辨识时间和辨识的模式。其中，输入信号的选择必须保证在辨识时间内使对象的动态特性被持续激励，从谱分析的角度看，就是要求输入信号的频谱必须覆盖对象的频谱；另一方面，输入信号的选择还应尽量提高辨识模型的精度，即具有较好的“优良性”。关于最优输入信号的设计问题，可以参考有关文献。从工程应用的角度出发，输入信号的选择还应考虑下面3点要求：

（1） 输入信号的功率或幅度不宜过大，以免对象进入非线性工作区；同时也应避免输入信号的幅值过小，否则将影响辨识的精度；

（2） 输入信号对过程的“净扰动”要小，即正、负向扰动的机会要尽量均等；

（3） 工程上容易实现，成本低。

系统辨识实验设计需要完成的另一项工作是采样时间的选择，这将直接影响到辨识的精度。采样时间的选择一般应遵循如下的原则：

（1） 满足采样定理，即采样频率不低于信号截止频率的2倍；

（2） 与模型最终应用时的采样时间尽可能保持一致，并且尽量考虑辨识算法；控制算法的计算速度和执行机构、检测元件的响应速度等问题；

（3） 采样时间过大或过小都不利于获得良好的辨识效果，在工程应用中选择采样时间通常采用的检验公式为

$$T_0=T_{95}/(5\sim15)$$

其中，T_0表示采样时间，T_{95}是过程阶跃响应达到稳态值95%需要的调节时间。

在系统辨识实验设计中还涉及到辨识模式的选择问题，即选择开环辨识或闭环辨识以及选择在线辨识或离线辨识。在开环辨识中，对象不存在反馈控制作用；而在闭环辨识中，对象处于闭环控制回路中。离线辨识是指对象的输入输出数据已经全部获得，对辨识计算不要求实时性；在线辨识则在过程对象的实际运行中完成，辨识结果具有实时性。

• 模型结构辨识

模型结构辨识包括模型类和模型结构参数的确定两部分内容。模型类的确定主要根据经验对实际对象的特性进行一定程度上的假设，如对象的模型是线性的还是非线性的、是参数模型还是非参数模型等。在确定模型类之后，就可根据对象的输入输出数据，按照一定的辨识算法确定模型结构参数。例如，对于如下的以差分方程表示的参数模型类

$$A(z^{-1})z(k)=z^{-d}B(z^{-1})u(k)+e(k)$$

其中，

$$A(z^{-1})=1+a_1z^{-1}+\dots+a_nz^{-n}$$
$$B(z^{-1})=b_1z^{-1}+b_2z^{-2}+\dots+b_nz^{-n}$$

对应上述模型类的模型结构参数为阶次 n 和纯时延 d。

• 模型参数辨识

在模型结构确定后，需要进行模型参数辨识。模型参数的辨识方法主要包括各种参数估计方法，如最小二乘法等。最小二乘法的基本结果有两种形式，一种是经典的一次完成算法；另一种是递推算法。最小二乘法及其各种改进算法如增广最小二乘法、广义最小二乘法等在系统辨识中具有重要的地位，是模型参数辨识的基本工具。最小二乘法考虑如下的输入输出关系：

$$z(k)=h^T(k)\theta+n(k)$$

其中，$z(k)$为对象的输出向量，$h(k)$是可观测的数据向量，θ为对象模型参数，$n(k)$为零均值的随机噪声。参数估计的优化指标为

$$J(\theta)=\sum[z(k)-h^T(k)\theta]^2$$

通过使 J=min 得到θ的估计值$\hat{\theta}$，$\hat{\theta}$称为θ的最小二乘估计值。

• 模型检验

模型检验是系统辨识不可缺少的步骤之一。模型检验的方法主要有：

(1) 利用在不同时间区间内采集的输入输出数据，分别进行对象模型辨识，如果各个模型的特性基本相同，则说明辨识结果是可靠的；

(2) 利用两组不同的数据分别得到辨识模型，并分别计算它们的损失函数；然后将两组数据交叉使用，再计算各自的损失函数，如果对应的损失函数没有显著变化，则说明辨识模型是可靠的；

(3) 增加辨识中使用的数据长度，如果损失函数不再显著下降，则模型是可靠的；

(4) 检验模型与对象输出残差序列$\{e(k)\}$的白色性，如果残差序列可以看作零均值的白噪声序列，则认为辨识模型是可靠的。

1.1.2 常用的模型类

作为系统辨识的三个基本要素之一，模型类的选择往往决定能否有效地建立对象的辨识模型。在 Matlab 系统辨识工具箱中提供了对多种模型类的支持，包括非参数模型类中的脉冲响应模型、参数模型类中的 ARX 模型、ARMAX 模型、BJ 模型和状态空间模型等。下面对这些模型类进行简要介绍，更进一步的介绍参见后续章节的函数说明。

1 非参数模型类

在非参数模型类中主要包括脉冲响应模型和频域描述模型。

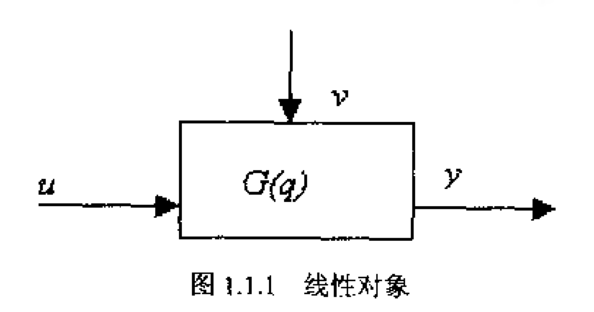


图 1.1.1 线性对象

在图 1.1.1 中，u 为输入，y 为输出，v 为外界噪声信号。对象的输入输出关系可以表示如下：

$$y(t)=G(q)u(t)+v(t)$$

$$G(q)=\sum_{k=1}^{\infty}g_k q^{-k}$$

其中，q 为时间平移算子。序列$\{g_k\}$称为对象的脉冲响应模型。

外界噪声信号 v 具有的频率特性设为 $\Phi_v(\omega)$，其计算公式为：

$$\Phi_v(\omega)=\sum_{\tau=-\infty}^{+\infty}R(\tau)e^{-j\omega\tau}$$

$$R(\tau)=Ev(t)v(t-\tau)$$

对 $G(q)$作傅氏变换得到 $G(e^{j\omega})$，称为对象的频率函数。对象频率函数 $G(e^{j\omega})$和噪声频率特性 $\Phi_v(\omega)$统称为对象的频域描述模型。这两种模型类都不能表示为对象的有限参数模型形式，因此称为非参数模型。

2 参数模型类

参数模型类是指利用有限参数来表示对象的模型，在系统辨识工具箱中支持的参数模型类有 ARX 模型、ARMAX 模型、BJ 模型、输出误差模型和状态空间模型。

- ARX 模型

ARX 模型具有如下的形式：

$$A(q)y(t)=q^{-nk}B(q)u(t)+e(t)$$
$$A(q)=1+a_1q^{-1}+a_2q^{-2}+\ldots+a_{na}q^{-na}$$
$$B(q)=b_1+b_2q^{-1}+\ldots+b_{nb}q^{-nb+1}$$

其中，*na* 和 *nb* 分别为相应多项式的阶次，*nk* 为对象的纯时延，*e(t)*为零均值白噪声。

- ARMAX 模型

ARMAX 模型具有如下的形式：

$$A(q)y(t)=q^{-nk}B(q)u(t)+C(q)e(t)$$
$$A(q)=1+a_1q^{-1}+a_2q^{-2}+\ldots+a_{na}q^{-na}$$
$$B(q)=b_1+b_2q^{-1}+\ldots+b_{nb}q^{-nb+1}$$
$$C(q)=1+c_1q^{-1}+\ldots+c_{nc}q^{-nc}$$

其中，*na*、*nb* 和 *nc* 分别为相应多项式的阶次，*nk* 为对象的纯时延，*e(t)*为零均值白噪声。

- Box-Jenkins 模型

Box-Jenkins 模型简称 BJ 模型，具有下面的形式：

$$y(t)=\frac{B(q)}{F(q)}u(t-nk)+\frac{C(q)}{D(q)}e(t)$$
$$B(q)=b_1+b_2q^{-1}+\ldots+b_{nb}q^{-nb+1}$$
$$F(q)=1+f_1q^{-1}+\ldots+f_{nf}q^{-nf}$$
$$C(q)=1+c_1q^{-1}+\ldots+c_{nc}q^{-nc}$$
$$D(q)=1+d_1q^{-1}+\ldots+d_{nd}q^{-nd}$$

其中，*nf*、*nb*、*nc* 和 *nd* 分别为相应多项式的阶次，*nk* 为对象的纯时延，*e(t)*为零均值白噪声。

- 输出误差模型

输出误差模型具有下面的形式：

$$y(t)=\frac{B(q)}{F(q)}u(t-nk)+e(t)$$
$$B(q)=b_1+b_2q^{-1}+\ldots+b_{nb}q^{-nb+1}$$
$$F(q)=1+f_1q^{-1}+\ldots+f_{nf}q^{-nf}$$

其中，*nf* 和 *nb* 分别为相应多项式的阶次，*nk* 为对象的纯时延，*e(t)*为零均值白噪声。

以上四种模型都可用如下的一般参数模型表示：

$$A(q)y(t)=\frac{B(q)}{F(q)}u(t-nk)+\frac{C(q)}{D(q)}e(t)$$

- 状态空间模型

状态空间模型适于描述多变量系统，其形式如下：

$$x(t+1)=Ax(t)+Bu(t)$$
$$y(t)=Cx(t)+Du(t)+v(t)$$

其中，*A*、*B*、*C* 和 *D* 为状态空间模型的系数矩阵，*v(t)*为外界噪声信号。

1.2 模型类的建立和转换

系统辨识工具箱提供的模型类建立和转换函数，如表 1.2.1 所示。

表 1.2.1 模型类建立和转换函数

函数名称	功 能	函数名称	功 能
arx2th	定义 ARX 模型类	idmodred	模型降阶
canform	生成正则模型形式	thc2thd	将连续系统模型转换为离散模型
mf2th	根据 M 文件生成任意线性模型结构	thd2thc	将离散模型转换为连续模型
modstruc	定义状态空间模型类	th2arx	Theta 模型格式转换为 ARX 参数
ms2th	生成线性状态空间模型	th2ff	Theta 模型格式转换为频域函数和谱
poly2th	生成由分子分母多项式定义的输入输出模型类	th2par	Theta 模型格式转换为估计参数和方差
fixpar	固定模型中的参数	th2poly	Theta 模型格式转换为传递函数多项式
sett	设置采样时间	th2ss	Theta 模型格式转换为状态空间模型
ss2th	将状态空间模型转换为参数正则模型形式	th2tf	Theta 模型格式转换为传递函数
thinit	选择或随机初始化参数值	th2zp	Theta 模型格式转换为零极点形式
unfixpar	允许对已被固定的参数进行估计		

1.2.1 模型建立函数

在系统辨识工具箱中提供了一种通用的参数模型结构和参数的存储与表示形式，即 Theta 模型格式。该格式以矩阵的形式表示、存储各种参数模型的模型结构和参数（包括固定值参数和估计参数），并能够方便地与各种参数模型进行相互转换。

1 Theta 模型格式

作为一种系统辨识工具箱中通用的参数模型格式，Theta 模型的定义可以分为两种类型，即基于输入输出表示的 Theta 模型和基于状态空间表示的 Theta 模型。基于输入输出的 Theta 模型可以对应各种输入输出参数模型，如 ARX、ARMAX、BJ 模型等；基于状态空间表示的 Theta 模型则与连续或离散状态空间参数模型对应。这两种 Theta 模型格式都以矩阵形式存储有关模型结构和参数的信息，但模型信息数据的组织不同。基于输入输出的 Theta 模型格式针对如下的线性多输入单输出模型结构：

$$A(q)y(t)=\frac{B_1(q)}{F_1(q)}u_1(t-nk_1)+\cdots+\frac{B_n(q)}{F_n(q)}u_{nu}(t-nk_{nu})+\frac{C(q)}{D(q)}e(t)$$

其中，$A(q)$、$B_i(q)$、$F_i(q)$（i=1,2,...,n）、$C(q)$和$D(q)$为平移算子 q 的多项式，其阶次分别为 na、nbi、nfi（i=1,2,...,n）、nc 和 nd，nu 为系统的输入变量个数。

设 n 为所有多项式的阶次之和，令 $r=\max(n,7,6+3nu)$，则系统的输入输出 Theta 模型格式为如下定义的 $(3+n)\times r$ 矩阵

(1) 矩阵的第一行为估计方差，采样时间 T，输入个数 nu，各个多项式的阶次 na、nb、nc、nd、nf 和 nk；

(2) 第二行为最终预测误差 FPE(Akaike)，模型格式生成的日期、时间和命令；

(3) 第三行为估计参数的向量，即 A、B、C、D 和 F 的系数（包括首 1 或首 0 系数）；

(4) 第四到第 $3+n$ 行为估计的方差矩阵；

(5) 对于连续系统模型，该矩阵可能增加第 $n+4$ 行，包含系统的死区时间。

对于基于状态空间表示的 Theta 模型格式，有一个 M 文件与其对应，调用格式为

$$[A, B, C, D, K, X0] = \mathrm{mfname}(par, T, aux)$$

其中，aux 为 nr×nc 参数矩阵。设估计参数的数目为 n，M 文件的名称长度为 r，输出个数为 ny，则 Theta 模型格式对应的矩阵维数为：(3+n+nr+ny)×max{n,ny,nc,7+r}，其定义如下：

(1) 第一行包括采样时间、输入输出维数、估计参数个数等信息；

(2) 第二行为最终预测误差 FPE(Akaike)，模型格式生成的日期、时间和命令，其中第二行的第 8 个元素定义为：

1——连续时间状态空间模型；

2——离散时间状态空间模型；

3——多变量 ARX 模型；

4——用户定义的离散时间状态空间模型；

5——用户定义的连续时间状态空间模型。

(3) 第三行为参数估计值；

(4) 第 4 到 3+n 行为参数的估计方差；

(5) 第 4＋n 到 3+n+nr 行为矩阵 aux；

(6) 第 4+n+nr 到 3+n+nr+ny 行包含了模型新息的方差矩阵。

2 模型建立函数

下面对系统辨识工具箱的模型建立函数进行介绍。

- arx2th

功能：由 ARX 多项式建立 Theta 模型矩阵。

格式：th = arx2th(A,B,ny,nu)

th = arx2th(A,B,ny,nu,lam,T)

说明：多输入多输出 ARX 模型具有如下的形式：

$$y(t)+A_1 y(t-1)+\ldots A_{na} y(t-na)=B_0 u(t)+B_1 u(t-1)+\ldots B_{nb} u(t-nb)+e(t)$$

其中 A_i（$i=1,2,\ldots,na$）和 B_j（$j=1,2,\ldots,nb$）分别为 ny×ny 和 ny×nu 的矩阵，ny、nu 分别为输出和输入的个数。

函数的输入参数定义为：

A——多项式$A(q)$的系数向量；

B——多项式$B(q)$的系数向量；

ny——输出的个数；

nu——输入的个数；

lam——可选输入，用于指定噪声源$e(t)$的方差矩阵，缺省值为单位阵；

T——可选输入，指定采样时间，缺省值为1。

输出参数定义为：

th——ARX 模型的 Theta 模型格式。

举例：

下面的 Matlab 程序建立 ARX 的 Theta 模型格式，然后在添加测量噪声后对系统模型进行仿真和辨识。

```
A1 = [-1.5 0.1;-0.2 1.5];
A2 = [0.7 -0.3;0.1 0.7];
B1 = [1;-1];
B2 = [0.5;1.2];
th0 = arx2th([eye(2) A1 A2],[[0;0],B1 B2],2,1);
u = idinput(300);
e = randn(300,2);
y = idsim([u e],th0);
th = arx([y u],[[2 2;2 2],[2;2],[1;1]]);
```

- canform

功能：定义多变量状态空间正则形式的模型结构。

格式：ms = canform(orders,nu)

ms = canform(orders,nu,dkx)

说明：该函数与函数 ms2th 配合使用，用于生成对象的状态空间模型 Theta 格式。其中 canform 的输出矩阵 ms 作为 ms2th 的输入参数。

输入参数定义为：

orders——维数与输出个数相同的行向量，该向量的各个元素用于定义对应输出的延迟；

nu——输入变量的个数；

dkx——决定状态空间模型的 D，K 和状态变量初值是否作为估计参数，D、K在状态空间模型中的定义如下：

$$\dot{x}(t) = Ax(t) + Bu(t) + Ke(t)$$
$$y(t) = Cx(t) + Du(t) + e(t)$$

dkx=[d,k,x]中的任一元素为1，则表示对应的参数矩阵 D、K 或 $X0$ 作为估计参数；若为0，则对应的参数矩阵固定为0。

输出参数定义为：

ms——状态空间模型结构矩阵，作为函数 ms2th 的输入。

举例:

```
ps = [2 1 3];
ms = canform(ps,2);
th = ms2th(ms,'c',ones(1,18+12+18)*NaN);
[A,B,C,D,K] = th2ss(th)
```

- **ms2th**

功能：建立基于状态空间表示的 Theta 模型格式。

格式：th = ms2th(ms)

th = ms2th(ms,cd,parval,lambda,T)

说明：该函数在利用 canform 构造状态空间模型结构的基础上，建立基于状态空间表示的 Theta 模型格式。

输入参数定义为：

ms——函数 canform 输出的正则状态空间模型结构；

cd——可选参数，为连续或离散系统的标志。当 cd='czoh'时，表示连续系统经过零阶采样保持的模型；当 cd='cfoh'时，表示连续系统经过一阶采样保持的模型；当 cd='d'时，表示为离散系统模型；

parval——估计参数（自由参数）的初始值向量，对应参数的初始值可用于参数估计或仿真和分析研究，缺省值均为 0；该向量中各个参数的排序按照系统状态空间矩阵 A、B、C、D、K 和 X0 的顺序，在每个矩阵内的估计参数则按行依次排列；

lambda——噪声源（新息）$e(t)$的方差矩阵，缺省值为单位矩阵；

T——采样间隔，该采样间隔被同时用于参数估计和仿真预测，因此当模型为连续系统模型时，采样间隔仍然要求大于 0。

举例：定义如下的连续时间状态空间模型结构。

$$\dot{x} = \begin{bmatrix} \theta_1 & 0 \\ 0 & \theta_2 \end{bmatrix} x + \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} u$$

$$y = [1 \quad 1] x + e$$

估计参数的初始值为

θ=[-0.2,-0.3,2,4]

完成上述功能并进行参数估计的 Matlab 程序如下：

```
A = [NaN,0;0,NaN];
B = [NaN;NaN];
C = [1,1];
ms = modstruc(A,B,C,0,[0;0]);
th = ms2th(ms,'c',[-0.2,-0.3,2, 4]);
th = pem(z,th);
```

- modstruc

功能：定义具有未知参数的状态空间模型结构。

格式：ms = modstruc(A,B,C,D,K)

ms = modstruc(A,B,C,D,K,X0)

说明：该函数定义如下形式的具有未知参数的状态空间模型结构：

$$\dot{x}(t) = A(\theta)x(t) + B(\theta)u(t) + K(\theta)e(t)$$
$$x(0) = x_0(\theta)$$
$$y(t) = C(\theta)x(t) + D(\theta)u(t) + e(t)$$

输入参数定义为：

A,B,C,D,K,X0——具有未知参数的状态空间模型矩阵，其中的未知参数用 NaN 表示。

输出参数定义为：

ms——状态空间模型结构矩阵，作为函数 ms2th 的输入。

举例：

```
A = [NaN,0;0,NaN];
B = [NaN;NaN];
C = [1,1];
D = 0;
K = [0;0];
ms = modstruc(A,B,C,D,K)
th = ms2th(ms,'c');
```

- poly2th

功能：构造基于一般的输入输出表示的 Theta 模型格式。

格式：th = poly2th(A,B)

th = poly2th(A,B,C,D,F,lam,T)

说明：该函数构造具有如下形式的多输入单输出系统 Theta 模型格式：

$$A(q)y(t) = \frac{B_1(q)}{F_1(q)}u_1(t-nk_1)+\ldots+\frac{B_n(q)}{F_n(q)}u_{nu}(t-nk_{nu})+\frac{C(q)}{D(q)}e(t)$$

输入参数定义为：

A，B，C，D，F——多项式 $A(q),B(q),C(q),D(q),F(q)$的系数向量或矩阵。对于单输入系统，上述参数均为向量，其中 A，C，D，F 都以首 1 系数为第一个元素，向量 B 则包含 0 元素以表示系统纯时延的大小；对于多输入系统，B 和 F 为矩阵形式，每一行对应一个输入的多项式系数；对于时间序列，B 和 F 为空矩阵。

lam——白噪声序列的方差。

T——采样时间，若 T<0，则表示为连续系统。

当参数 C、D、F 和 T 省略时，表示对应值为 1。

输出参数定义为：

th——系统的输入输出 Theta 模型格式。

举例：

```
A = [1 -1.5 0.7];
B = [0 1 0.5];
C = [1 -1 0.2];
th0 = poly2th(A,B,C);
```

- fixpar

功能：固定 Theta 模型的某些参数。

格式：thn = fixpar(tho,matrix)

thn = fixpar(tho,matrix,elements,parval)

说明：该函数通过给原有 Theta 模型中的估计参数赋值来生成一个新的 Theta 模型。原有的 Theta 模型可由函数 arx、arx2th、ms2th 等函数构造。

输入参数定义为：

tho——原有的 Theta 模型格式；

matrix——指定被赋值的估计参数所在的系统参数向量或矩阵，对于状态空间 Theta 模型格式，matrix 的取值可以为：'A'、'B'、'C'、'D'、'K'和'X0'；对于 ARX 模型的 Theta 模型格式（参见函数 arx2th），matrix 的取值可以为：'A1'、'A2'、... 'B0'、'B1',...；

Elements——用于指定被赋值的估计参数在相应矩阵的位置；该参数为一个具有两列的矩阵，其中第一列用于指定估计参数所在的行数，第二列用于指定估计参数所在的列数；

Parval——用于指定估计参数被赋予的值，如果被省略，则指定的估计参数将被赋予初始值或当前估计值。

输出参数定义为：

thn——对原有 Theta 模型中部分参数赋值后得到的新的 Theta 模型。

举例：

```
th = arx(z,[2 3 0]);
th = fixpar(th,'B2',[1,2],1);
th = pem(z,th);
```

- sett

功能：设置采样时间间隔。

格式：modn = sett(mod,T)

说明：该函数可以直接改变 Theta 模型格式或频域模型格式的采样时间间隔。输入参数定义为：

mod——系统的 Theta 模型格式或频域模型格式；

T——新的采样时间间隔。

输出参数定义为：

modn ——改变采样时间间隔后新的系统模型。

举例：

```
th = armax(z,nn);
th = sett(th,T);
```

- ss2th

功能：生成参数的正则状态空间 Theta 模型格式。

格式：THS = ss2th(TH)

THS = ss2th(TH,orders，dkx)

说明：该函数能够将任意的Theta模型格式转换为参数的正则状态空间Theta模型格式，有关正则状态空间模型结构的说明可以参见函数canform。

输入参数定义为：

TH——任意的Theta模型格式；

orders——维数与输出个数相同的行向量，称为伪可观性指标向量(pseudo-observability indices)，该向量的各个元素用于定义对应输出的延迟；

dkx——决定状态空间模型的 D，K 和状态变量初值是否作为估计参数，D、K 在状态空间模型中的定义如下：

$$\dot{x}(t) = Ax(t) + Bu(t) + Ke(t)$$
$$y(t) = Cx(t) + Du(t) + e(t)$$

dkx=[d,k,x]中的任一元素为 1，则表示对应的参数矩阵 D、K 或 X0 作为估计参数；若为 0，则对应的参数矩阵固定为 0。

输出参数定义为：

THS——参数的正则状态空间 Theta 模型格式，该模型的确定参数取值与 TH 相同。

- thinit

功能：随机设置估计参数的初始值。

格式：th = thinit(th0)

th = thinit(th0,R,pars,sp)

说明：该函数随机设置系统 Theta 模型格式的估计参数，输入参数定义为：

th0——对象原有的 Theta 模型格式；

R——随机化估计参数的方差向量；

Pars——随机化估计参数的均值向量；

Sp——指定参数初始化对系统稳定性和预测器稳定性的要求，当 sp='s'时，仅要求参数的初始化保证系统的稳定性；当 sp='p'时，仅要求参数的初始化保证预测器的稳定性；当 sp='b'时，要求参数的初始化同时保证系统和预测

器的稳定性。

上述输入参数中，R、pars 和 sp 为可选参数，缺省值分别为：

R——全 1 的向量；

Pars——th0 中的参数值；

Sp——缺省值为'p'。

输出参数定义为：

th——经过参数随机初始化的系统 Theta 模型格式。

- unfixpar

功能：将经过函数 fixpar 赋值的固定参数作为估计参数。

格式：thn = unfixpar(tho,matrix)

thn = unfixpar(tho,matrix,elements)

说明：输入输出参数的定义参见函数 fixpar。

1.2.2 模型转换函数

在 Matlab 系统辨识工具箱中提供了若干函数用于各种模型格式之间的转换。下面对这些函数的使用方法进行介绍。

1 idmodred

功能：对系统的 Theta 模型进行降阶。

格式：THRED = idmodred(TH)

THRED = idmodred(TH,ORDER,OE)

说明：该函数能够对任意的系统 Theta 模型格式进行降阶处理，降阶后得到的新的 Theta 模型通常表示为无估计参数的状态空间形式。降阶算法采用了控制系统工具箱的函数 balreal 和 modred(参见本书第 2 章)。

输入参数定义为：

TH——系统未降阶的 Theta 模型；

ORDER——可选参数，用于指定要求的降阶模型阶数，缺省值为空；

OE——用于指定是否生成输出误差降阶模型，当 OE ＝ 'oe'时，生成一个 Kalman 滤波增益为 0 的输出误差降阶模型；当 OE 为其他值或省略时，噪声模型与对象模型同时被降阶。

输出参数定义为：

THRED——降阶后的对象 Theta 模型。

举例：

```
A = [1 -0.5 0.7]; B = [0  1 0.5];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,randn(300,1)],th0);
```

```
TH = arx([y u],[4,4,1]);
THRED = idmodred(TH,3);
```

2 thc2thd

功能：将连续时间 Theta 模型转换为离散时间 Theta 模型。

格式：thd = thc2thd(thc,T)

说明：输入参数定义为：

thc——连续时间 Theta 模型；

T——采样时间。

输出参数定义为：

thd——离散时间 Theta 模型。

举例：下面的 Matlab 程序将系统的连续时间 Theta 模型转换为离散时间 Theta 模型，并绘制离散模型的零极点图（如图 1.2.1 所示）。

```
thc = poly2th(1,1,1,1,[1 1 0],1,-1);
thd = thc2thd(thc,0.5);
zpplot(th2zp(thd))
```

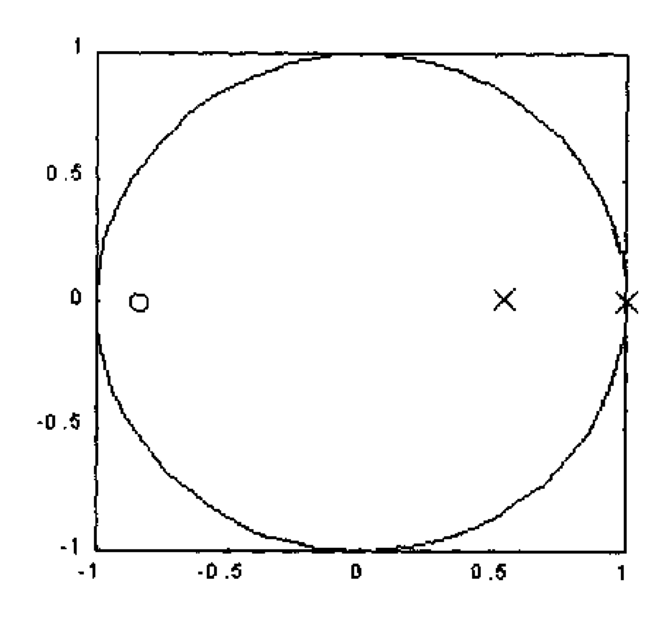


图 1.2.1　离散模型的零极点图

3 thd2thc

功能：离散时间 Theta 模型转换为连续时间 Theta 模型。

格式：thc = thd2thc(thd)

thc = thd2thc(thd,delay,NoP)

说明：输入参数定义为：

thd——离散时间 Theta 模型；

delay——用于在离散系统具有纯时延时，决定是否对纯时延进行逼近；delay

有两种取值，即

• 'del'：对纯时延用有限维连续系统进行逼近；

• 'nodel'：不对纯时延进行逼近，此时在变换过程中不考虑纯时延，而在变换结束后将纯时延直接添加到连续系统模型中，为缺省值；

NoP——决定是否对方差矩阵进行变换，NoP 为 1 时不对方差矩阵进行变换。

输出参数定义为：

thc——离散时间 Theta 模型。

举例：下面的 Matlab 程序将离散 Theta 模型转换为连续 Theta 模型，并比较两者的波特图。

```
A = [1 -1.5 0.7];
B = [0 1 0.5];
C = [1 -1 0.2];
th = poly2th(A,B,C);
gd = th2ff(th);
thc = thd2thc(th);
gc = th2ff(thc);
bodeplot([gd, gc],3)
```

4 th2arx

功能：从 Theta 模型格式中获得 ARX 模型参数。

格式：[A,B] = th2arx(th)

[A,B,dA,dB] = th2arx(th)

说明：输入参数 th 为对象的 Theta 模型格式，输出参数 A、B 对应 ARX 的参数向量。ARX 模型的形式为

$$y(t)+A_1y(t-1)+...+A_{na}y(t-na)=B_0u(t)+B_1u(t-1)+...+B_{nb}u(t-nb)$$

对应的输出参数向量 A，B 为

$$A=[I,A_1,A_2,...A_{na}]$$

$$B=[B_0,B_1,...B_{nb}]$$

输出参数 dA,dB 分别为 A，B 的标准差。

5 th2ff

功能：计算 Theta 模型的频率响应和标准差。

格式：[g,phiv] = th2ff(th)

[g,phiv] = th2ff(th,ku,w,ky)

说明：输入参数定义为：

th——对象的 Theta 模型；

ku,w,ky 为可选参数

w——用于计算频率响应的频率向量；对于连续时间模型其缺省值为

$w = \log space(\log 10(pi/abs(T)/100), \log 10(10*pi/abs(T)), 128)$

对于离散时间模型，w 的缺省值为

$w = [1:128]/128*pi/T$

ku,ky——用于指定多变量系统的输入输出变量。

输出参数定义为：

g——对象的频率函数 $G(e^{j\omega})$；

phiv——噪声的谱估计函数 $\Phi_v(\omega)$。

举例：下面的 Matlab 程序对给定数据进行 ARMAX 建模后，由函数 th2ff 计算对应的频率函数和噪声谱估计，并与直接通过输入输出数据的计算结果比较。通过函数 th2ff 计算得到的频率响应波特图如图 1.2.2 所示。直接由输入输出数据计算得到的频率响应波特图如图 1.2.3 所示。

```
B = [0 1 0.5];
C = [1 -1 0.2];
D = [1 1.5 0.7];
F = [1 -0.5 0.7];
th0 = poly2th(1,B,C,D,F,0.1);
e - randn(200,1);
u = idinput(200);
y = idsim([u e],th0)
z = [y u];
th = armax(z,[2 2 2 1]);
gp = th2ff(th);
bodeplot(gp)
gs=spa(z)
bodeplot(gs)
```

6 th2par

功能：从 Theta 模型中提取模型参数。

格式：[par,P,lam] = th2par(th)

说明：该函数从 Theta 模型中得到系统状态空间模型的估计参数。输入参数 th 为系统的 Theta 模型。输出参数定义为：

par——系统状态空间模型的估计参数向量，par 的形式为

$$par = \{a_1, a_2, \ldots a_{na}, b_1^1, \ldots b_{nb1}^1, b_1^2, \ldots b_{nb2}^2, \ldots b_{nbnu}^{nu}, c_1, c_2, \ldots c_{nc}, d_1, d_2, \ldots d_{nd}, f_1^1, f_2^1, \ldots f_{nf1}^1, f_1^2, \ldots f_{nfnu}^{nu}\}$$

P——参数的方差矩阵；

Lam——新息的方差。

7 th2poly

功能：将 Theta 模型转换为多项式模型形式。

格式：[A,B,C,D,F,LAM,T] = th2poly(th)

说明：输入参数 th 为对象的 Theta 模型,输出参数为对象输入输出模型的多项式系数向量或矩阵。对象的输入输出模型具有如下形式：

$$A(q)y(t)=\frac{B_1(q)}{F_1(q)}u_1(t-nk_1)+\ldots\frac{B_{nu}(q)}{F_{nu}(q)}u_{nu}(t-nk_{nu})+\frac{C(q)}{D(q)}e(t)$$

参数 LAM 为噪声的方差矩阵，T 为采样时间间隔。

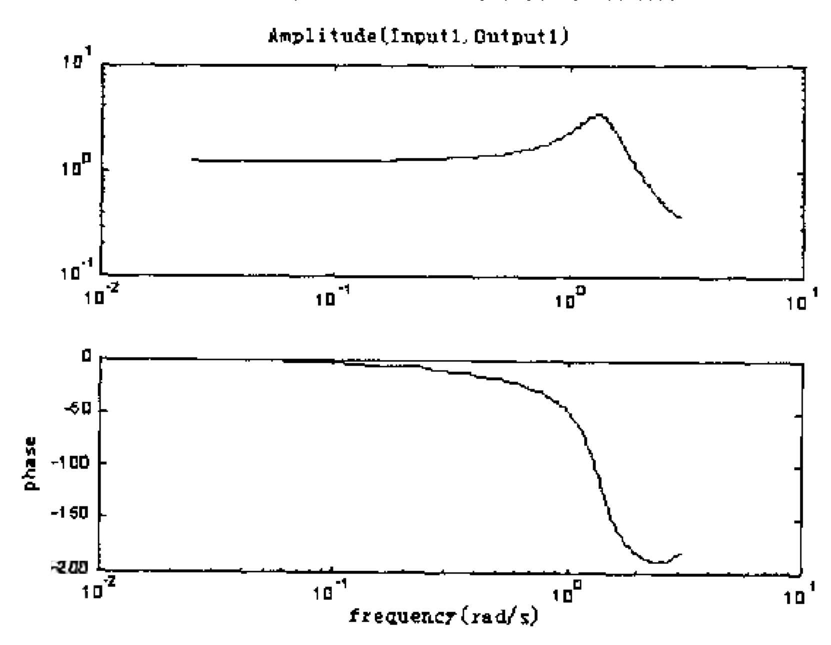


图 1.2.2　由函数 th2ff 计算得到的频率波特图

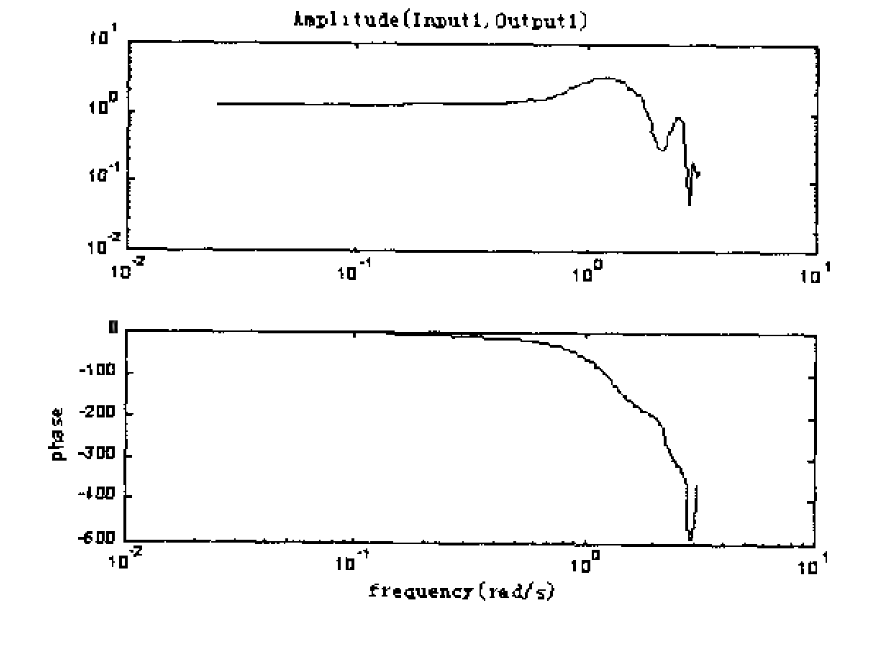


图 1.2.3　由谱分析计算得到的频率响应波特图

8 th2ss

功能：将 Theta 模型转换为状态空间模型形式。

格式：[A,B,C,D,K,X0] = th2ss(th)

[A,B,C,D,K,X0,dA,dB,dC,dD,dK,dX0] = th2ss(th)

说明：输入参数 th 为对象的 Theta 模型,输出参数 A，B，C，D，K，X0 为对应状态空间模型的系数矩阵。状态空间(连续时间)模型具有下面的形式：

$$\dot{x}(t) = A(\theta)x(t) + B(\theta)u(t) + K(\theta)e(t)$$
$$x(0) = x_0(\theta)$$
$$y(t) = C(\theta)x(t) + D(\theta)u(t) + e(t)$$

离散时间状态空间模型具有类似的形式：

$$x(t+1) = A(\theta)x(t) + B(\theta)u(t) + K(\theta)e(t)$$
$$x(0) = x_0(\theta)$$
$$y(t) = C(\theta)x(t) + D(\theta)u(t) + e(t)$$

输出参数 dA,dB,dC,dD,dK,dX0 分别对应参数 A，B，C，D，K，X0 的标准差。当对象的 Theta 模型为基于状态空间表示的模型格式时，输出参数即为对应的模型参数；当对象的 Theta 模型为基于输入输出表示的模型格式时，得到的是对象的观测器标准形状态空间模型。

9 th2tf

功能：将 Theta 模型转换为传递函数模型形式。

格式：[num,den] = th2tf(th)

[num,den] = th2tf(th,iu)

说明：输入参数 th 为对象的 Theta 模型，iu 用于指定输入变量；输出参数 num 和 den 分别为对象传递函数模型的分子和分母多项式系数构成的矩阵，其中 num 的第 k 行对应对象第 k 个输出的传递函数分子多项式系数。

若要得到第 k 个噪声源到输出的传递函数多项式系数，可令 iu=-k。

10 th2zp

功能：计算对象 Theta 模型的零极点和稳态增益。

格式：[zepo,k] = th2zp(th)

[zepo,k] = th2zp(th,ku,ky,thresh)

说明：输入参数 th 为对象的 Theta 模型。参数 ku,ky,thresh 为可选参数，定义为：

ku——指定输入变量；

ky——指定输出变量；

thresh——计算无穷零点的阈值，当零点的绝对值大于该阈值时，则作为无穷零点处理。

输出参数定义为：

zepo——包含对象零极点的矩阵；

k——对象的稳态增益。

举例：下面的 Matlab 程序分别计算对象不同阶数的 ARX 模型的零极点，并绘制各自的零极点图。对象的二阶 ARX 模型的零极点图如图 1.2.4 所示，三阶 AR 模型的零极点如图 1.2.5 所示。

```
B = [0 1 0.5];
C = [1 -1 0.2];
D = [1 1.5 0.7];
F = [1 -0.5 0.7];
th0 = poly2th(1,B,C,D,F,0.1);
e = randn(200,1);
u = idinput(200);
y = idsim([u e],th0);
z = [y u];
th2 = arx(z,[2 2 1]);
th3 = arx(z,[3 3 1]);
zp2 = th2zp(th2);
zp3 = th2zp(th3);
zpplot(zpform(zp2);
zpplot(zpform(zp3));
```

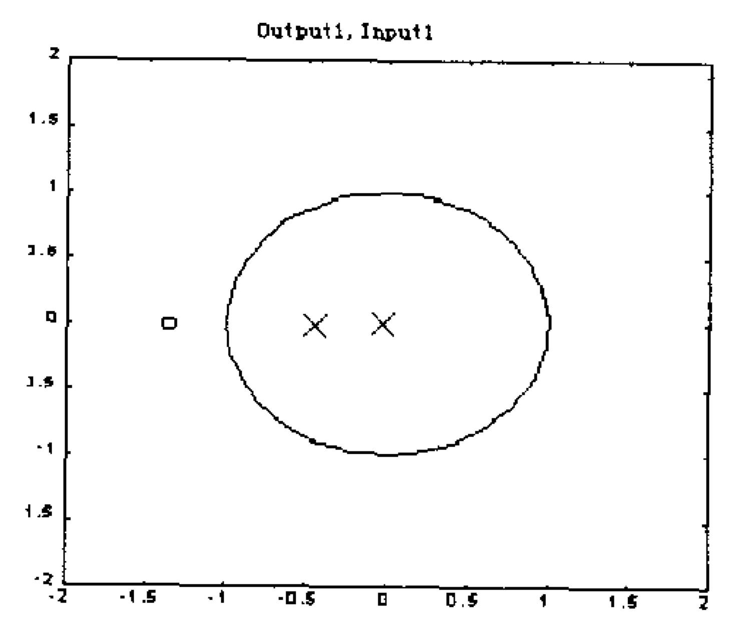


图1.2.4　二阶ARX模型的零极点图

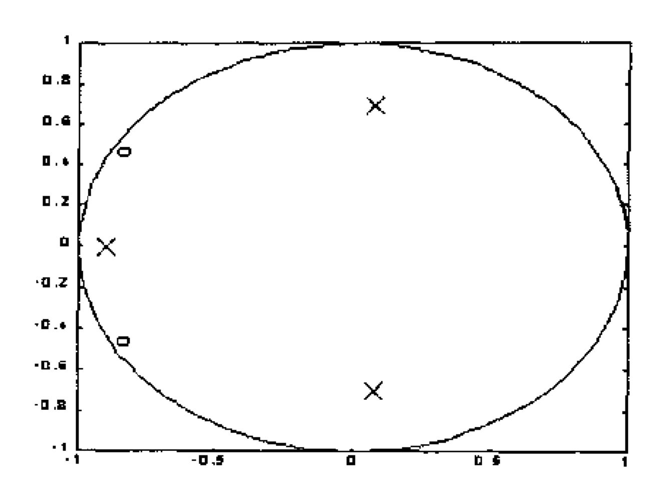


图 1.2.5　三阶 ARX 模型的零极点图

1.3　非参数模型辨识

在 Matlab 系统辨识工具箱中支持两种非参数模型格式，即脉冲响应模型和频域描述模型。有关这两种模型的辨识函数，如表 1.3.1 所示。

表 1.3.1　非参数模型辨识函数

函数名称	功　能
covf	估计时间序列的协方差函数
cra	采用相关分析方法估计对象的脉冲响应和方差函数
etfe	直接采用 Fourier 分析估计对象的频域谱和传递函数
spa	采用谱分析技术估计对象的频谱和传递函数

1.3.1　时间序列协方差函数的估计

对于 n 维的时间序列$\{z(t)\}$，$z_i(t)$的协方差函数（或称为自相关函数）定义为

$$R_i(\tau)=Ez_i(t)z_i(t+\tau)$$

$z_i(t)$与 $z_j(t)$的互协方差函数定义为

$$R_{ij}(\tau)=Ez_i(t)z_j(t+\tau)$$

当时间序列$\{z(t)\}$的长度为 N 时，可以得到各个分量的协方差函数和互协方差函数的估计为：

$$\hat{R}_{ii}(\tau)=\frac{1}{N}\sum_{t=1}^{N}z_i(t)z_i(t+\tau)$$

$$\hat{R}_{ij}(\tau)=\frac{1}{N}\sum_{t=1}^{N}z_i(t)z_j(t+\tau)$$

函数 covf 用于完成上述协方差函数和互协方差函数估计的计算。

- covf

功能：估计时间序列的协方差函数。

格式：R = covf(z,M)

R = covf(z,M,maxsize)

说明：输入参数定义为：

z——n 维时间序列{z(t)}对应的矩阵；

M——计算协方差函数和互协方差函数的最大延迟 τ_{max}；

Maxsize——可选参数，用于控制计算要求的存储器容量。

输出参数定义为：

R——n 维时间序列{z(t)}协方差和互协方差函数的估计结果，为 $n^2\times M$ 维矩阵，其元素定义如下：

$$R(i+(j-1)n,k+1)=R_{ij}(k)=\frac{1}{N}\sum_{t=1}^{N}z_i(t)z_j(t+k)$$

1.3.2　对象脉冲响应的辨识

在获得对象的输入输出数据后，可以利用相关分析方法估计对象的脉冲响应模型。设对象的输入输出序列为$\{u(t),y(t)\}$，当输入为白噪声序列时，其自相关函数(或协方差函数)为

$$R_u(\tau)=Eu(t)u(t+\tau)=\begin{cases}\lambda &,\tau=0\\0 &,\tau\neq 0\end{cases}$$

输入输出的互相关函数为

$$R_{uy}(\tau)=Eu(t)y(t+\tau)=\lambda\, g(\tau)$$

其中 $g(\tau)$即为对象的脉冲响应。

对于有限长度的输入输出序列（设长度为 N），对象脉冲响应的估计由下式计算：

$$\hat{g}(\tau)=\frac{1}{N\lambda}\sum_{t=1}^{N}y(t+\tau)u(t)$$

当输入不是白噪声信号时，可以先对输入进行白化处理，即通过一个白化滤波器 $L(q)$，

得到的新输入信号 $u_F(t)=L(q)u(t)$近似为白噪声序列。将经过白化处理的输入信号叠加到对象的输入端，同时对输出进行同样的白化处理，则可利用前面的结果进行对象的相关分析，估计对象的脉冲响应。

函数 cra 用于实现上述的相关分析计算，其使用方法说明如下。

- cra

功能：采用相关分析方法估计对象的脉冲响应。

格式：cra(z)

[ir,R,cl] = cra(z,M,na,plot)

cra(R)

说明：该函数仅能用于单输入单输出对象的相关分析。

输入参数定义为：

z——输入输出数据，z=[y u]；

M——指定计算协方差（自相关）函数的最大延迟，即协方差函数计算的时间范围为[-M,M]，同时脉冲响应计算的时间范围为[0,M]，缺省值 M=20；

na——白化滤波器的阶次；

plot——指定是否绘制脉冲响应曲线或协方差函数曲线，当 plot=0，则不绘制曲线；当 plot=1，绘制对象的脉冲响应曲线；当 plot=2，绘制对象的输入输出自相关函数曲线。

输出参数定义为：

ir——对象脉冲响应的估计；

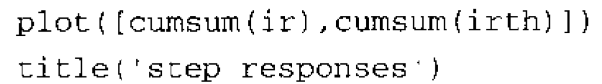
R——输入输出的协方差函数矩阵；

cl——具有 99%信念因子的脉冲响应估计。

当采用 cra(R)的调用格式时，则直接绘制对象输入输出协方差函数曲线。

举例：下面的 Matlab 程序分别对一个二阶对象进行 ARX 建模和相关分析，并比较两种方法得到的脉冲响应和阶跃响应，如图 1.3.1 所示。

```
A = [1 -0.5 0.7]; B = [0  1 0.5];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,randn(300,1)],th0);
z=[y u];
ir = cra(z);
th = arx(z,[2 2 1]);
imp = [1;zeros(19,1)];
irth = idsim(imp,th);
subplot(211)
plot([ir irth])
title('impulse responses')
subplot(212)
```



```
plot([cumsum(ir),cumsum(irth)])
title('step responses')
```

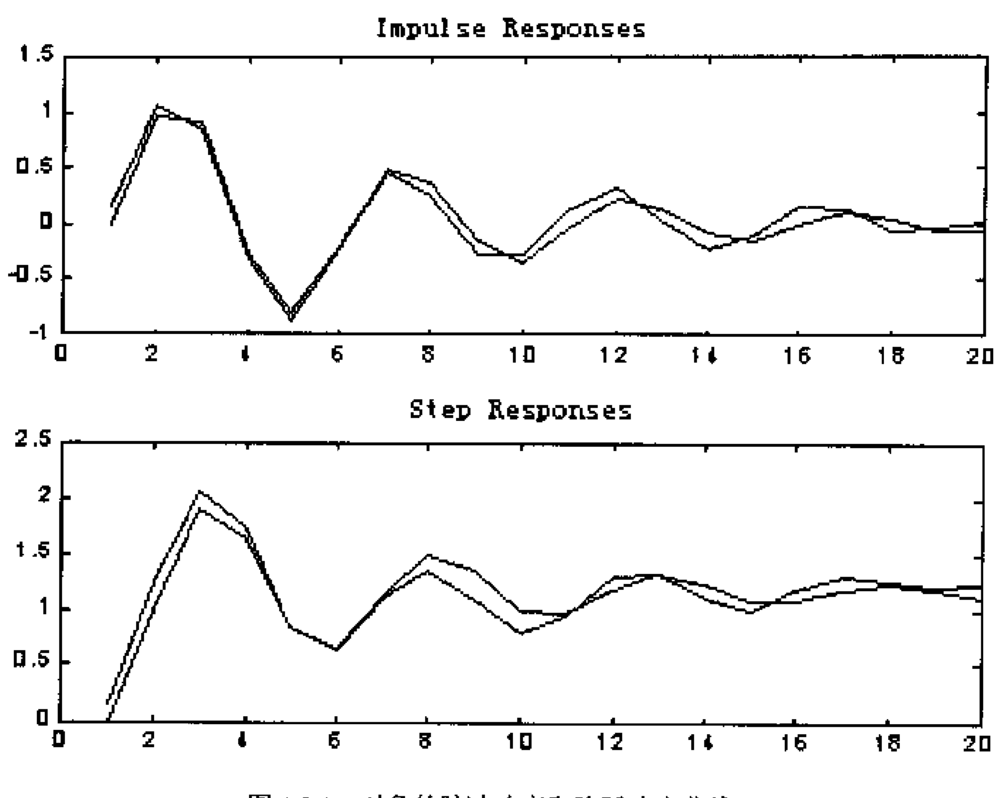


图 1.3.1　对象的脉冲响应和阶跃响应曲线

1.3.3　控制对象频谱和传递函数的辨识

设对象的输入输出关系为

$$y(t)=G(q)u(t)+v(t)$$

其中，$u(t)$为输入信号，$v(t)$为噪声信号。当输入与噪声不相关时，输出$y(t)$的自相关函数为

$$R_y(\tau)=Ey(t)y(t+\tau)=E(G(q)u(t)+v(t))(G(q)u(t+\tau)+v(t+\tau))$$

$$=E(G^2(q)u(t)u(t+\tau)+v(t)v(t+\tau))$$

对上式进行 Fourier 变换，得到对应的频谱函数关系

$$\Phi_y(\omega)=\left|G(e^{i\omega})\right|^2\Phi_u(\omega)+\Phi_v(\omega)$$

输入与输出的互相关函数

$$R_{yu}(\tau)=Ey(t)u(t+\tau)=EG(q)u(t)u(t+\tau)$$

对上式进行 Fourier 变换，得到频谱函数关系

$$\Phi_{yu}(\omega)=G(e^{i\omega})\Phi_u(\omega)$$

当获得对象的输入输出数据后，可对输入输出的自相关函数以及输入输出的互相关函数进行估计：

$$\hat{R}_u(\tau)=\frac{1}{N}\sum_{t=1}^{N}u(t)u(t+\tau)$$

$$\hat{R}_y(\tau)=\frac{1}{N}\sum_{t=1}^{N}y(t)y(t+\tau)$$

$$\hat{R}_{yu}(\tau)=\frac{1}{N}\sum_{t=1}^{N}u(t)y(t+\tau)$$

分别对上面三式进行 Fourier 变换，可以得到对应的频谱函数的估计：

$$\hat{\Phi}_u(\omega)=\sum_{\tau=-M}^{M}\hat{R}_u(\tau)W_M(\tau)e^{-i\omega\tau}$$

$$\hat{\Phi}_y(\omega)=\sum_{\tau=-M}^{M}\hat{R}_y(\tau)W_M(\tau)e^{-i\omega\tau}$$

$$\hat{\Phi}_{yu}(\omega)=\sum_{\tau=-M}^{M}\hat{R}_{yu}(\tau)W_M(\tau)e^{-i\omega\tau}$$

其中， $W_M(\tau)$为滞后窗函数，M 为滞后窗的宽度。

在获得上述结果后，可用下式计算对象的频率响应函数估计和噪声的频谱函数估计：

$$\hat{G}_N(e^{i\omega})=\frac{\hat{\Phi}_{yu}(\omega)}{\hat{\Phi}_u(\varpi)}$$

$$\hat{\Phi}_v(e^{i\omega})=\hat{\Phi}_y(\omega)-\frac{\left|\hat{\Phi}_{yu}(\omega)\right|^2}{\hat{\Phi}_u(\omega)}$$

上面对对象频率响应和噪声频谱函数的估计方法称为频谱分析方法。函数 spa 可用于完成频谱分析的计算过程，其用法说明如下。

1 spa

功能： 利用频谱分析方法估计对象的频率响应和噪声频谱。

格式： [g,phiv] = spa(z)

[g,phiv,z_spe] = spa(z,M,w,maxsize,T)

说明： 输入参数定义为：

z——对象的输入输出数据向量或时间序列向量，z=[y u]或 z=y，y、u 分别为输出和输入数据构成的列向量；

M——可选参数，滞后窗的宽度，缺省值定义为

M=min(30,leagth(z)/10);

W——可选参数，为频率向量形式，用于指定计算频谱函数的频率点，缺省值为

W=[1:128]/128*pi/T;

T——采用时间间隔；

Maxsize——可选参数，用于计算过程中的存储与速度控制。

输出参数定义为：

g——对象的频率响应函数估计$\hat{G}(e^{i\omega})$；

phiv——噪声的频谱估计$\hat{\Phi}_v(\omega)$。

举例：

```
A = [1 -0.5 0.7];
B = [0  1 0.5];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,randn(300,1)],th0);
z=[y u];
w = logspace(-2,pi,128);
[g,phiv] = spa(z,[],w);
bodeplot([g phiv])
```

对象的频率响应估计和噪声频谱估计的波特图，分别如图 1.3.2 和 1.3.3 所示。

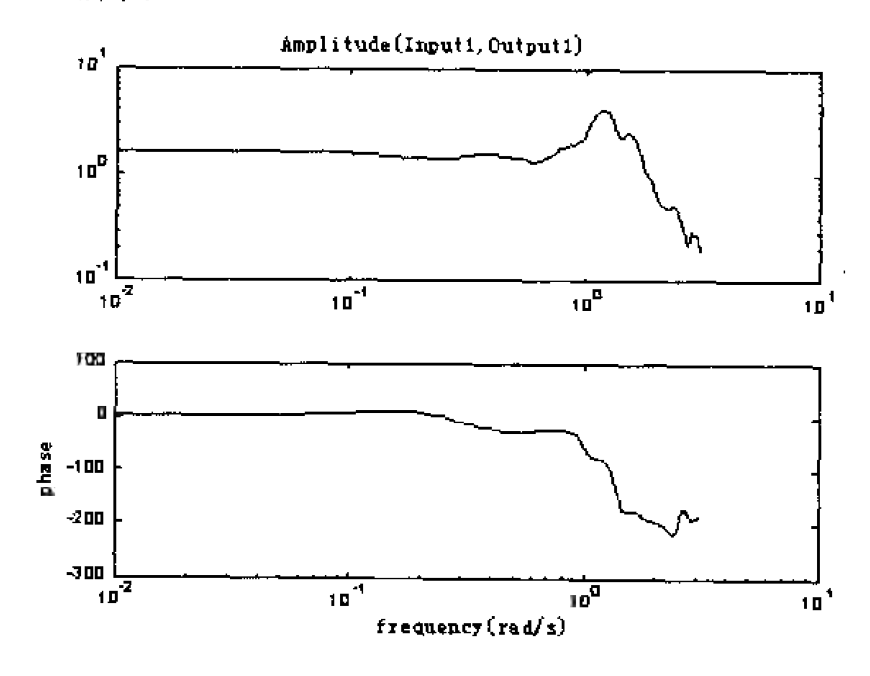


图 1.3.2　对象频率响应波特图

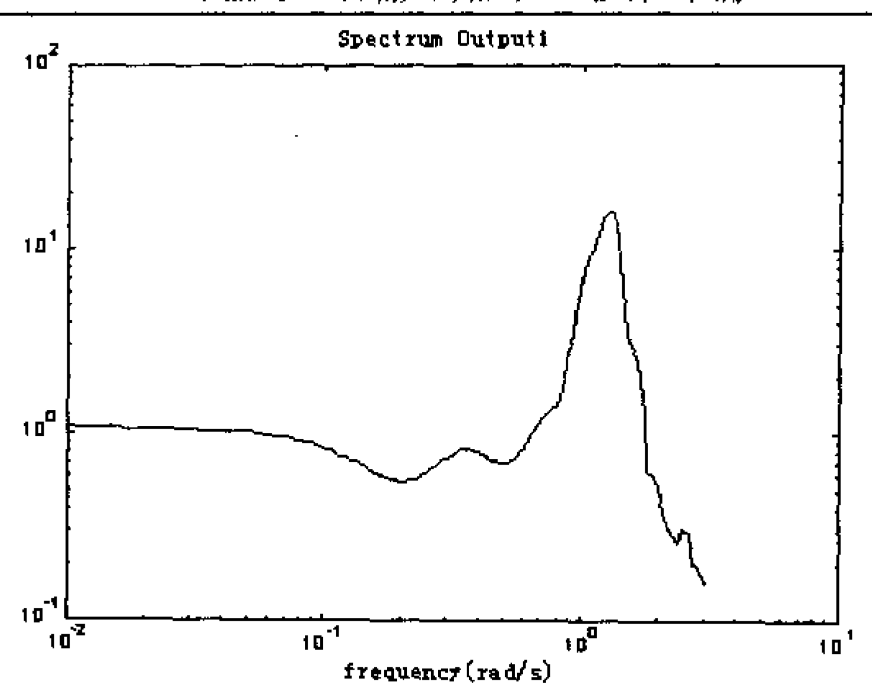


图 1.3.3　噪声频谱波特图

估计对象频率响应的方法除了上面介绍的频谱分析方法外，还可以直接对对象的输入输出数据进行 Fourier 变换，然后将输出的 Fourier 变换与输入的 Fourier 变换的比值作为对象频率响应的估计。函数 etfe 进行直接基于 Fourier 变换的频率响应估计。

2　etfe

功能：直接基于快速 Fourier 变换估计对象的频率响应。

格式：g = etfe(z)

g = etfe(z,M,N,T)

说明：输入参数定义为：

z——对象的输入输出数据向量或时间序列向量，z=[y u]或 z=y，y、u 分别为输出和输入数据构成的列向量；

M——可选参数，用于指定对频谱估计的平滑操作；

N，T——可选参数，用于指定计算频率响应的频率向量，W=[1:N]/N*pi/T。

输出参数定义为：

g——对象的频率响应函数。

举例：对给定的输入输出数据分别采用频谱分析方法和基于直接 Fourier 变换的方法估计对象的频率响应，并加以比较。其中图 1.3.4 所示为采用频谱分析方法得到的对象频率响应，图 1.3.5 所示为基于直接 Fourier 变换方法得到的对象频率响应。

```
A = [1 -0.5 0.7];
B = [0  1 0.5];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
```

```
y = idsim([u,randn(300,1)],th0);
z=[y u];
ge = etfe(z);
gs = spa(z);
bodeplot(ge)
bodeplot(gs)
```

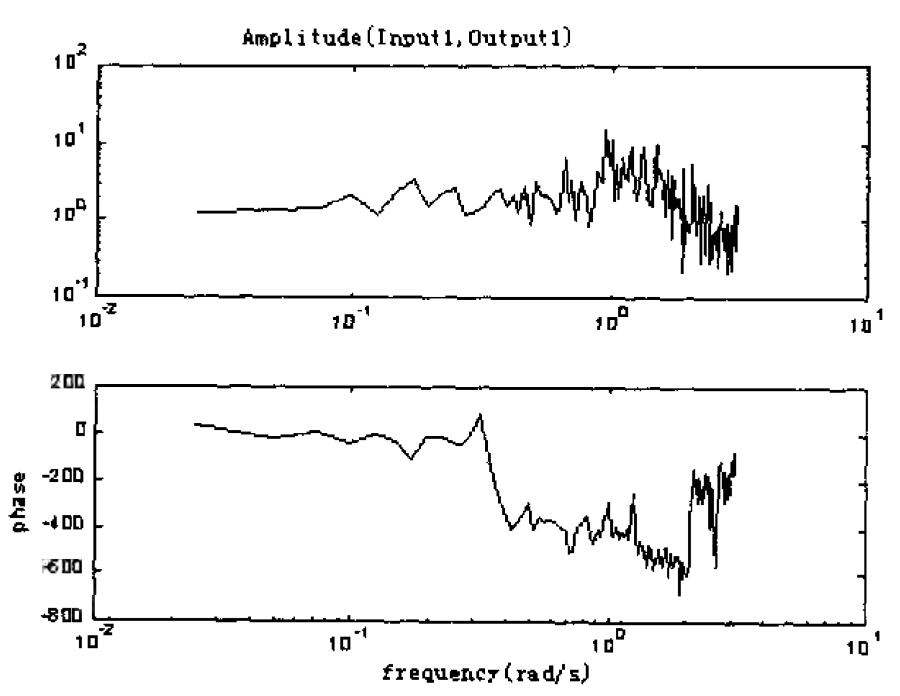


图 1.3.4　基于频谱分析方法得到的对象频率响应

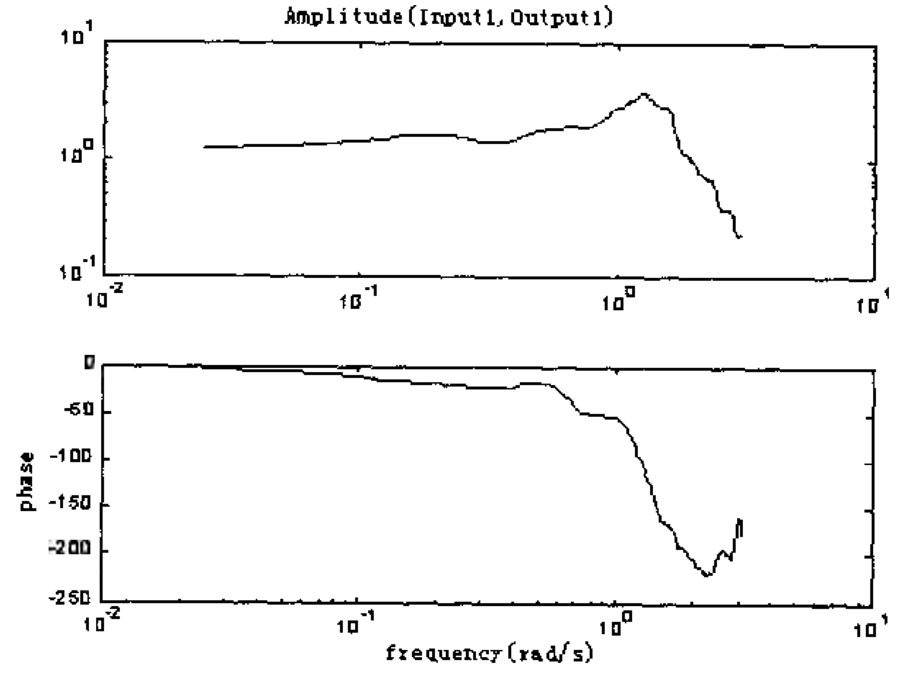


图 1.3.5　基于直接 Fourier 变换方法得到的对象频率响应

1.4 参数模型辨识

Matlab 系统辨识工具箱支持的参数模型类型，包括 AR/ARX、ARMAX、BJ 模型、状态空间模型和输出误差模型等。在该工具箱中都提供了相应的模型辨识函数，如表 1.4.1 所示。

表 1.4.1 参数模型辨识函数

函数名称	功能
ar	AR 模型辨识
armax	ARMAX 模型辨识
arx	ARX 模型辨识
bj	Box-Jenkins 模型辨识
canstart	估计多变量正则状态空间模型
ivar	采用辅助变量方法的 AR 模型辨识
ivx	采用辅助变量方法的 ARX 模型辨识
iv4	采用 4 步辅助变量方法估计 ARX 模型
oe	估计输出误差模型
n4sid	采用子空间方法估计状态空间模型
pem	估计一般的线性模型

1.4.1 AR 模型辨识

AR 模型具有如下的形式：

$$A(q)y(t)=e(t)$$

其中，$y(t)$为系统的输出，$e(t)$为白噪声输入信号。关于平移算子 q 的多项式 $A(q)$定义为

$$A(q)=1+a_1q^{-1}+a_2q^{-2}+\ldots+a_{na}q^{-na}$$

函数 ar 用于上述 AR 模型的辨识，其使用方法如下。

1 ar

功能：AR 模型辨识。

格式：th = ar(y,n)

[th,refl] = ar(y,n,approach,win,maxsize,T)

说明：输入参数定义为：

y——对象在白噪声作用下的输出；

n——AR 模型的阶次 na；

approach,win,maxsize,T 均为可选参数，其中

approach——用于指定参数估计的最小二乘类方法，approach 的取值可为：

• 'fb'：前向-后向方法为缺省方法，该方法以前向模型和后向模型的预测误差平方和为参数估计的极小化指标；

• 'ls'：标准的最小二乘方估计方法，以前向模型的预测误差平方和为极小化指标；

• 'yw'：采用 Yule-Walker 方法进行参数估计，该方法通过求解由采样方差构成的 Yule-Walker 方程获得参数的估计；

• 'burg'：采用 Burg 的基于网格的方法，该方法利用前向和后向平方预测误差的一致均值求解网格滤波器方程以得到参数的估计；

• 'gl'：采用几何网格方法进行参数估计，该方法利用前向和后向平方预测误差的几何均值，求解网格滤波器方程以得到参数的估计；

win——选择数据窗口的类型，win 的取值由以下几种定义：

• 'now'：无数据窗口，为缺省值；

• 'prw'：采用前向窗口，即测量开始点之前的 n 个输出数据被置 0，极小化指标的求和下限为 0 时刻，n 为对象 AR 模型阶次；

• 'pow'：采用后向窗口，即测量结束时刻后的 n 个输出数据点被置 0，极小化指标的求和上限为 N+n，N 为观测点个数，n 为对象 AR 模型阶次；

• 'ppw'：同时采用前向和后向窗口，即测量开始点之前的 n 个数据和测量结束时刻后的 n 个输出数据点均被置 0，这种数据窗口用于 Yule-Walker 参数估计方法。

参数 maxsize 和 T 用于指定计算过程的控制变量，都属于系统辨识工具箱使用的辅助变量 auxvar。辅助变量 auxvar 还包括 tol、iter_info、lim 和 maxiter。这些变量的定义为：

maxsize——用于指定计算过程中允许生成的矩阵的最大维数，缺省值的设定在文件 idmsize 中完成；

T——指定采样间隔，该采样间隔决定了绘制频率响应特性的频率尺度，同时也决定了连续模型转换为离散模型的采样周期，缺省值 T=1；

tol——指定 Gauss-Newton 迭代终止的范数条件，即当 Gauss-Newton 更新向量的范数小于 tol 时结束迭代过程；

lim——决定采用鲁棒估计技术的误差界限，当误差大于该界限时，则采用鲁棒估计技术，即将二次型指标改变为线性指标；缺省值为 lim=1.6；

Maxiter——指定迭代的最大次数，缺省值为 10；

Iter_info——计算结束后返回的迭代过程信息。

输出参数定义为：

th——AR 模型对应的 Theta 模型格式；

refl——当采用基于网格的方法（即 approach='burg'或'gl'）时，refl 用于返回反射系数和对应的损失函数。

函数 ar 仅用于尺度化时间序列，对于多尺度的序列可以采用函数 arx。

举例：对时间序列 $y(t)=\sin(t)+0.5e(t)$进行 AR 模型估计，并绘制频率响应波特图，其中采用缺省的参数估计方法得到的 AR 模型频谱，如图 1.4.1 所示。

```
y = sin([1:300]')+0.5*randn(300,1);
thb = ar(y,4,'burg');
thfb = ar(y,4);
sgb = th2ff(thb);
sfb = th2ff(thfb);
bodeplot(sgb)
```

上面介绍的基于最小二乘类参数估计的 AR 模型辨识方法，要求输入噪声为白噪声。当输入噪声为有色噪声时，则不能保证模型参数估值的无偏性和一致收敛性。为进行输入噪声为有色噪声的 AR 模型辨识，有关学者提出了辅助变量法来获得模型参数的无偏一致估计。

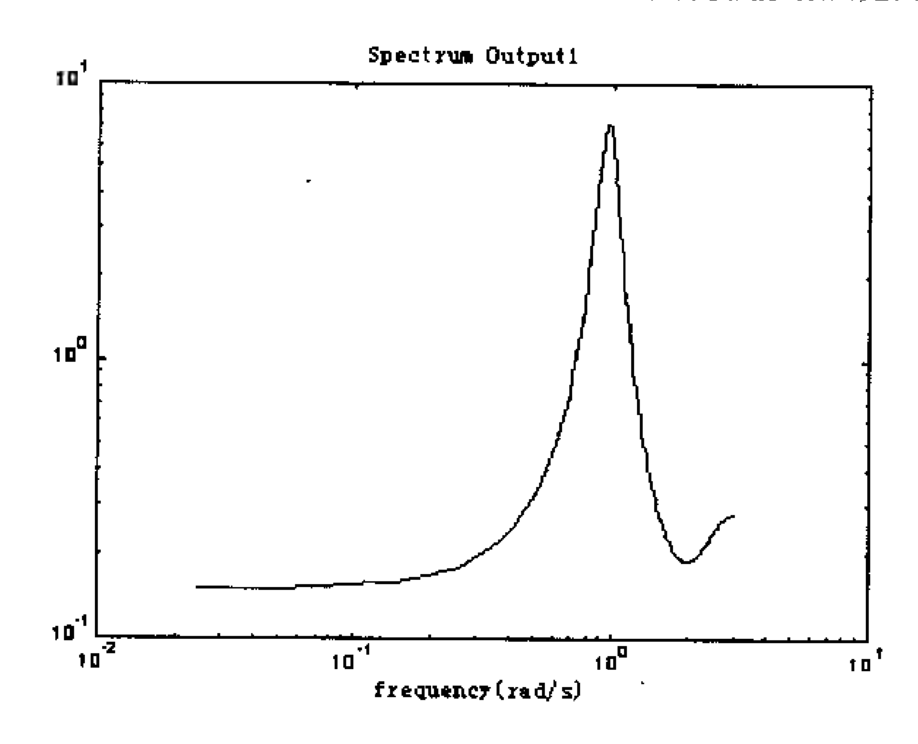


图 1.4.1　采用缺省参数估计方法的 AR 模型频谱

函数 ivar 用于基于最优辅助变量选择的 AR 模型辨识，该函数的使用方法说明如下。

2 ivar

功能：基于最优辅助变量选择的 AR 模型辨识。

格式：th = ivar(y,na)

th = ivar(y,na,nc,maxsize,T)

说明：该函数对如下的具有有色噪声输入的 AR 模型进行辨识

$$A(q)y(t)=v(t)$$

其中，$v(t)$为有色噪声，并且在参数估计过程中被假定为阶次为 nc 的滑动平均过程。

输入参数定义为：

y——输出序列，为列向量的形式；

na——指定 AR 模型的阶次；

nc——指定噪声特性的阶次，缺省值为 nc=na；

maxsize,T 为可选参数，用于指定计算过程中的控制参数，参见函数 ar。

输出参数定义为：

th——AR 模型对应的 Theta 模型格式。

举例：考虑如下的过程

$$y(t)=\sin(1.2t)+\sin(1.5t)+0.2v(t-1)+0.1v(t)$$

其输入噪声为有色噪声。下面分别利用最小二乘方估计和辅助变量方法，对上述过程进行 AR 模型辨识，得到的 AR 模型频谱分别如图 1.4.2 和 1.4.3 所示。

```
v=randn(501,1)
y = sin([1:500]'*1.2) + sin([1:500]'*1.5)
     + 0.2*v([1:500])+0.1*v([2:501]);
thiv = ivar(y,4);
thls = ar(y,4);
giv = th2ff(thiv);
gls = th2ff(thls);
bodeplot(gls)
bodeplot(giv)
```

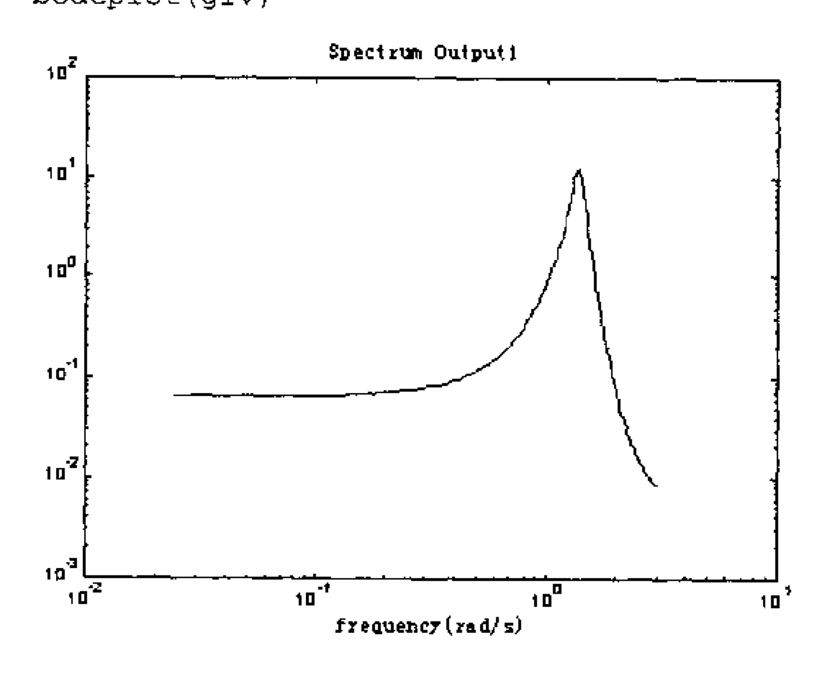


图 1.4.2　采用最小二乘方估计的 AR 模型频谱

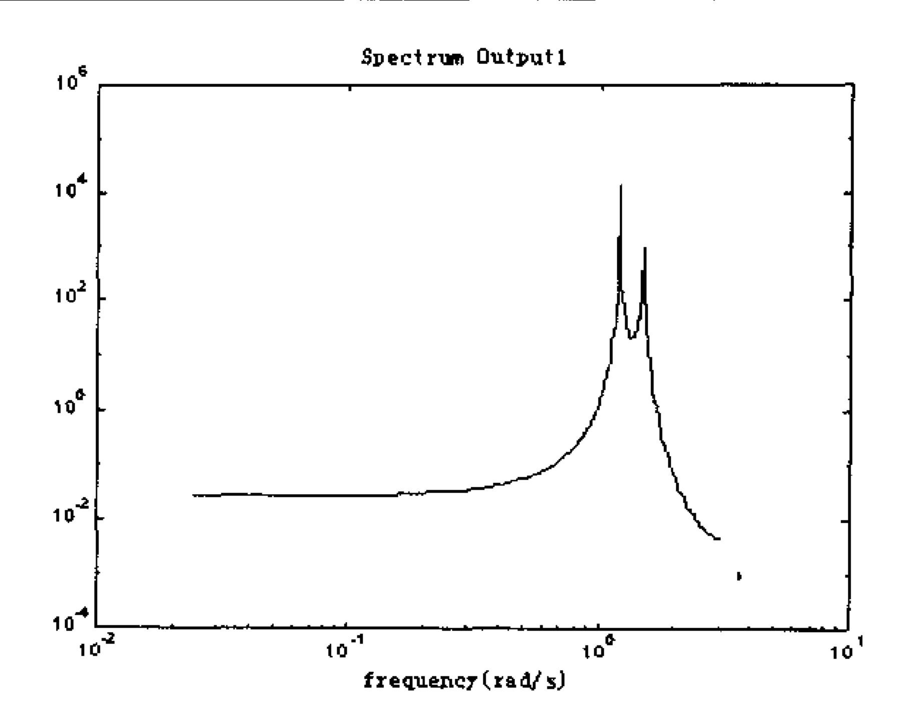


图 1.4.3 采用辅助变量方法获得的 AR 模型频谱

1.4.2 ARX 模型辨识

ARX 模型具有如下的形式：

$$A(q)y(t)=B(q)u(t-nk)+e(t)$$

其中，$A(q),B(q)$ 为关于平移算子 q 的多项式，定义为：

$$A(q)=1+a_1q^{-1}+a_2q^{-2}+...+a_{na}q^{-na}$$

$$B(q)=b_1+b_2q^{-1}+b_3q^{-2}+...+b_{nb}q^{-nb+1}$$

函数 arx 用于完成基于最小二乘方估计的 ARX 模型辨识。

1 arx

功能： 基于最小二乘方估计的 ARX 模型辨识。

格式： th = arx(z,nn)

th = arx(z,nn,maxsize,T)

说明： 输入参数定义为：

z——对象的输入输出数据矩阵，z=[y u]，其中 y 为对象输出数据向量，u 为对象输入数据向量，y、u 均为列向量的形式；

nn——用于指定 ARX 模型的阶次和纯时延大小，nn=[na nb nk]，对于多输入变量的情形，nb、nk 均为行向量形式；

参数 maxsize 和 T 用于指定计算过程的控制变量，参见函数 ar 关于辅助变量 auxvar 的说明。

输出参数定义为：

th——用 Theta 模型格式表示的 ARX 估计模型。

举例：下面的程序对一个二阶对象进行 ARX 模型辨识，并绘制 ARX 模型对应的频率响应波特图，如图 1.4.4 所示。

```
A = [1 -0.5 0.7]; B = [0 1 0.5];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,randn(300,1)],th0);
z = [y,u];
th = arx(z,[2 2 1]);
sth=th2ff(th)
bodeplot(sth)
```

与 AR 模型辨识的辅助变量方法相对应，ARX 模型的辨识也可采用辅助变量方法进行输入噪声不满足白噪声假设条件下的模型参数估计。在系统辨识工具箱中有两个类似的函数，都能用于基于辅助变量方法的 ARX 模型辨识。这两个函数是 iv4 和 ivx。其中，iv4 采用近似最优的 4 阶段辅助变量方法进行 ARX 模型辨识，ivx 则采用任意的辅助变量。

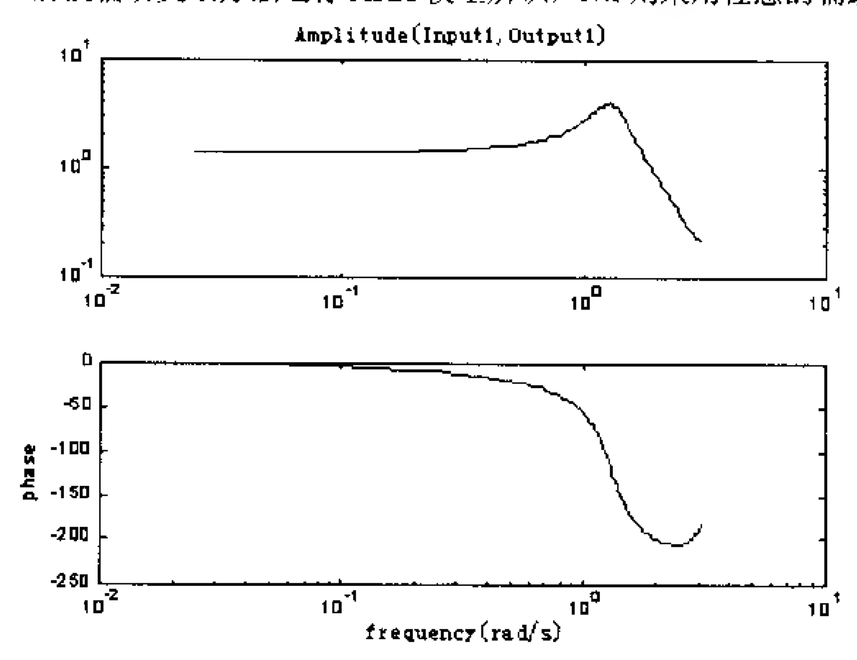


图 1.4.4　ARX 模型的频率响应波特图

2　iv4

功能：采用近似最优 4 阶段辅助变量方法的 ARX 模型辨识。

格式：th = iv4(z,nn)

th = iv4(z,nn,maxsize,T)

说明：输入参数定义为：

z——对象的输入输出数据矩阵，z=[y u]，其中 y 为对象输出数据向量，u 为对象输入数据向量，y、u 均为列向量的形式；

nn——用于指定 ARX 模型的阶次和纯时延大小，nn=[na nb nk]；

参数 maxsize 和 T 用于指定计算过程的控制变量，参见关于辅助变量 auxvar 的说明。

输出参数定义为：

th——用 Theta 模型格式表示的 ARX 估计模型。

举例：

```
v=randn(300,1)
A = [1 -0.5 0.7]; B = [0 1 0.5];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,v],th0);
z = [y,u];
thiv=iv4(z,[2 2 1]);
thiv=th2ff(thiv)
```

3　ivx

功能：采用任意辅助变量的 ARX 模型辨识。

格式：th = iv4(z,nn,x)

th = iv4(z,nn,x,maxsize,T)

说明：输入参数 z 和 nn 的定义参见函数 iv4；

x 定义为指定的辅助变量矩阵。

1.4.3　ARMAX 模型辨识

ARMAX 模型具有如下的形式：

$$A(q)y(t) = q^{-nk}B(q)u(t) + C(q)e(t)$$

对上述 ARMAX 模型参数辨识，通常采用的一种方法是预测误差方法，即通过极小化二次预测误差指标来获得参数的估计。优化的方法可以采用迭代 Gauss-Newton 算法等。系统辨识工具箱的函数 armax，用于完成基于预测误差方法的 ARMAX 模型辨识，其用法说明如下。

1　armax

功能：基于预测误差方法的 ARMAX 模型辨识。

格式：th = armax(z,nn)

th = armax(z,nn, 'trace')

[th, iter_info] = armax(z,nn,maxiter,tol,lim,maxsize,T, 'trace')

说明：输入参数定义为：

z——对象的输入输出数据矩阵，z=[y u]，其中 y 为对象输出数据向量，u 为对象输入数据向量，y、u 均为列向量的形式；

nn——用于指定 ARMAX 模型的阶次和纯时延大小；

nn=[na nb nc nk]，其中 na 为 AR 部分的阶次，nb 为 MA 部分的阶次，nc 为噪声特性的阶次，nk 为对象的纯时延；

'trace'——用于指定显示迭代优化过程的信息；

maxiter,tol,lim,maxsize,T——可选参数，均为系统表示工具箱使用的辅助变量，这些变量的定义参见函数 ar 的说明。

输出参数 th 为用 Theta 模型格式表示的 ARMAX 模型。

举例：下面的程序对一个二阶对象进行 ARMAX 建模，并绘制 ARMAX 模型的频率响应波特图，如图 1.4.5 所示。

```
v=randn(301,1)
v1=v(2:301)
v2=0.2.*v(1:300)
A = [1 -0.1 0.2]; B = [0 1 0.5];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,v2,v1],th0);
z = [y,u];
th=armax(z,[2 2 2 1]);
thf=th2ff(th)
bodeplot(thf)
```

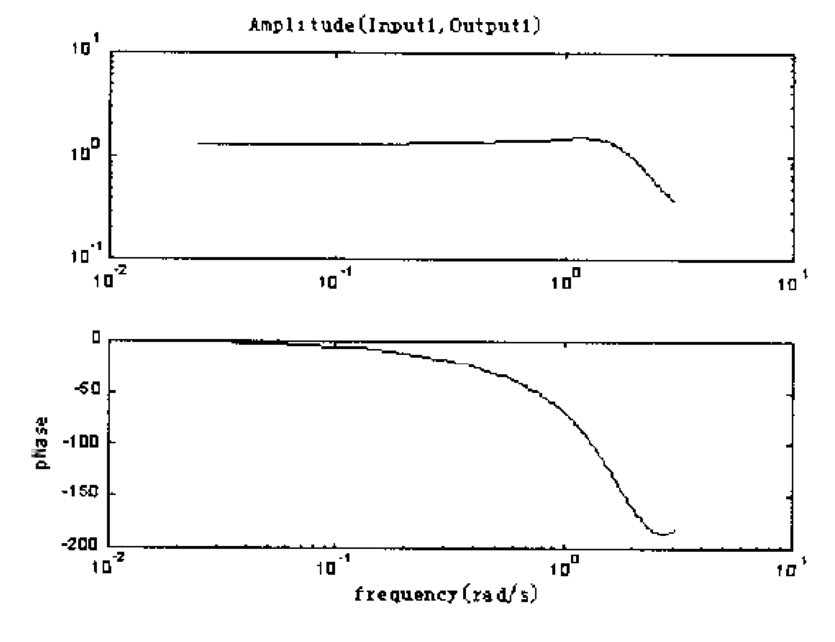


图 1.4.5　ARMAX 辨识模型的波形

1.4.4 输出误差模型与 BJ 模型辨识

输出误差模型具有如下的形式：

$$y(t)=\frac{B(q)}{F(q)}u(t-nk)+e(t)$$

BJ 模型即 Box-Jenkins 模型具有如下的形式：

$$y(t)=\frac{B(q)}{F(q)}u(t-nk)+\frac{C(q)}{D(q)}e(t)$$

函数 oe 采用预测误差方法进行输出误差模型辨识。

1 oe

功能：基于预测误差方法的输出误差模型辨识。

格式：th = oe(z,nn)

th = oe(z,nn, 'trace')

[th, iter_info] = oe(z,nn,maxiter,tol,lim,maxsize,T,'trace')

说明：输入参数定义为：

z——对象的输入输出数据矩阵，z=[y u]。其中 y 为对象输出数据向量，u 为对象输入数据向量，y、u 均为列向量的形式；

nn——用于指定 ARX 模型的阶次和纯时延大小，nn=[nb nf nk]。其中 nb 为多项式 B(q)的阶次，nf 为多项式 F(q)的阶次，nk 为对象的纯时延；

'trace'——用于指定显示迭代优化过程的信息；

其他参数的定义与函数 armax 相同。

举例：考虑与函数 armax 举例中相同的对象，对其进行输出误差模型辨识，并绘制输出误差模型的频率响应波特图，如图 1.4.6 所示。

```
v=randn(301,1)
v1=v(2:301)
v2=0.2.*v(1:300)
A = [1 -0.1 0.2];
B = [0 1 0.5];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,v2,v1],th0);
z = [y,u];
th=oe(z,[2 2 1])
thf=th2ff(th)
bodeplot(thf)
```

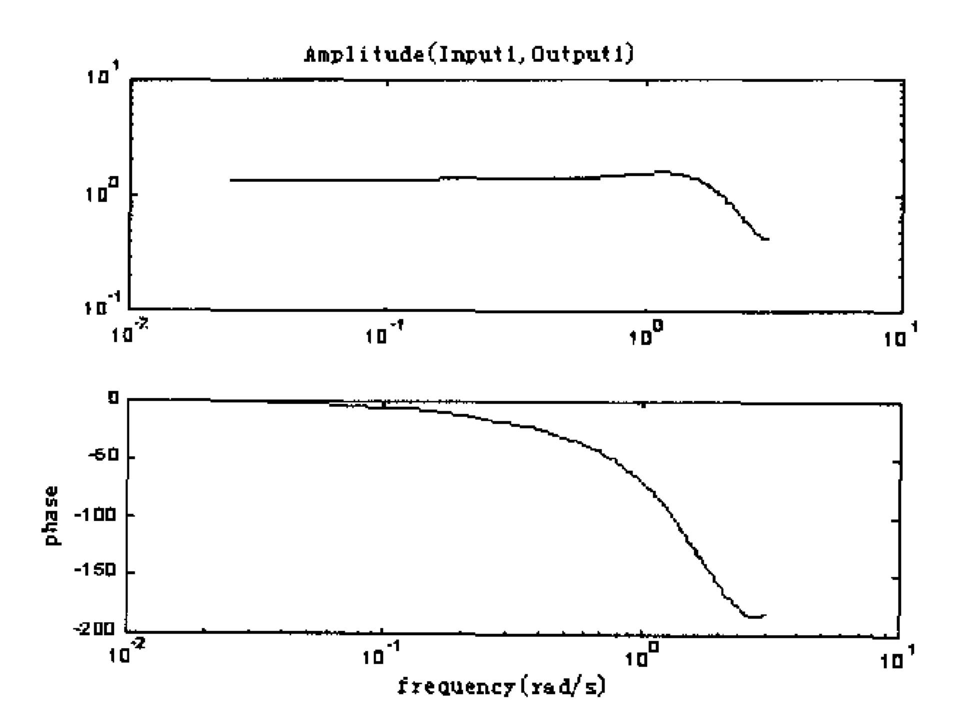


图 1.4.6 输出误差模型的波特图

用于 BJ 模型辨识的函数为 bj，其使用方法说明如下。

2 bj

功能：基于预测误差方法的 BJ 模型辨识。

格式：th = bj(z,nn)

th = bj(z,nn, 'trace')

[th, iter_info]= bj(z,nn,maxiter,tol,lim,maxsize,T, 'trace')

说明：输入参数定义为：

z——对象的输入输出数据矩阵，z=[y u]。其中 y 为对象输出数据向量，u 为对象输入数据向量，y、u 均为列向量的形式；

nn——用于指定 BJ 模型的阶次和纯时延，nn=[nb nc nd nf nk]或 thi。其中 nb 为多项式 B(q)的阶次，nf 为多项式 F(q)的阶次，nc、nd 分别为多项式 C(q)和 D(q)的阶次，nk 为对象的纯时延；thi 为标准的 Theta 模型格式，用于确定 BJ 模型各个多项式的阶次；

参数 maxiter,tol,lim,maxsize,T, 'trace'的定义，参见函数 ar。

举例：考虑具有如下模型的对象

$$y(t)=\frac{q^{-1}+0.5}{q^{-2}-1.5q^{-1}+0.7}u(t-nk)+\frac{q^{-2}-q^{-1}+0.2}{q^{-2}+1.5q^{-1}+0.7}e(t)$$

下面利用函数 bj 对该对象进行辨识

```
B = [0 1 0.5];
C = [1 -1 0.2];
```

```
D = [1 1.5 0.7];
F = [1 -1.5 0.7];
th0 = poly2th(1,B,C,D,F,0.1);
e = randn(200,1);
u = idinput(200);
y = idsim([u e],th0);
z = [y u];
thi = bj(z,[2 2 2 2 1],0);
th = bj(z,thi);
present(th)
```

辨识得到的各个多项式的参数估计值和标准差为:

```
B = 0      1.0001     0.4999
    0      0.0044     0.0063
F = 1.0000    -1.5005     0.7006
    0      0.0007     0.0005
C =  1.0000   -0.8238    0.1036
     0      0.0902     0.0900
D -1.0000    1.6034    0.7517
   0     0.0622    0.0604
```

%绘制辨识模型的波特图,如图 1.4.7 所示。

```
thf=th2ff(th)
bodeplot(thf)
```

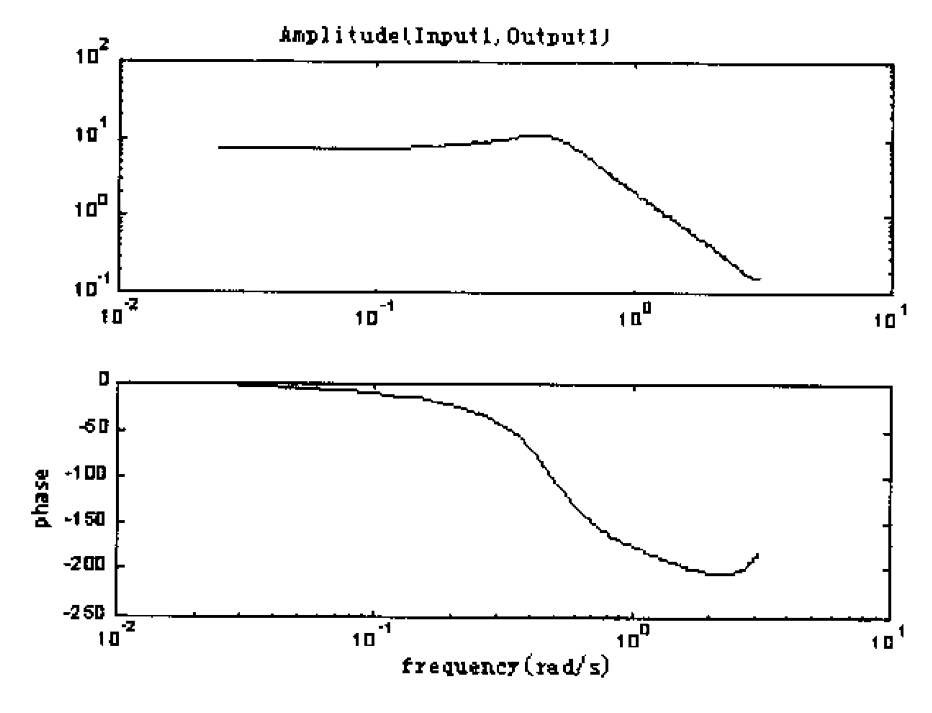


图 1.4.7 BJ 模型的频率波特图

1.4.5 状态空间模型辨识

函数 canstart 和 n4sid 分别用于多变量正则状态空间模型的辨识和基于子空间方法的状态空间模型辨识，使用方法说明如下。

1 canstart

功能：多变量正则状态空间模型辨识。

格式：th = canstart(z,orders,nu)

th = canstart(z,orders,nu,dkx)

说明：输入参数定义为：

z——对象的输入输出数据矩阵，z=[y u]，其中 y 为对象输出数据向量，u 为对象输入数据向量，y、u 均为列向量的形式；

orders——维数与输出个数相同的行向量，该向量的各个元素用于定义对应输出的延迟；

nu——输入变量的个数；

dkx——决定状态空间模型的 D、K 和状态变量初值是否作为估计参数，D、K 在状态空间模型中的定义为

$$\dot{x}(t) = Ax(t) + Bu(t) + Ke(t)$$
$$y(t) = Cx(t) + Du(t) + e(t)$$

dkx=[d,k,x]中的任一元素为 1，则表示对应的参数矩阵 D、K 或 X0 作为估计参数；若为 0，则对应的参数矩阵固定为 0。

输出参数 th 为以 Theta 模型格式表示的正则状态空间模型。

举例：

```
th=canstart(z,3,2)
```

2 n4sid

功能：基于子空间方法的状态空间模型辨识。

格式：TH = n4sid(z)

[TH,AO] = n4sid(z,order,ny,auxord,dkx,maxsize,T,'trace')

说明：该函数的输入参数定义为：

z——对象的输入输出数据矩阵，z=[y u]。其中 y 为对象输出数据向量，u 为对象输入数据向量，y、u 均为列向量的形式；对于时间序列，z=y；

order——以行向量的形式指定状态空间模型的阶次，缺省值为 order=[1:10]，即模型的阶次在 1 到 10 之间。函数 n4sid 对所有指定的阶次进行状态空间模型辨识，并绘图比较不同阶次模型的脉冲响应 Hankel 矩阵奇异值，由用户选择模型的阶次；

ny——输出的个数，缺省值为 1；

auxorder——辅助阶次，该阶次由子空间算法使用，作为预测时域的长度；auxorder 的取值应大于指定的阶次 order，缺省值为 1.2*order+3；辅助阶次的选择将对辨识模型的质量有重要影响；

dkx——决定状态空间模型的 D、K 和状态变量初值是否作为估计参数，D、K 在状态空间模型中的定义为：

$$\dot{x}(t) = Ax(t) + Bu(t) + Ke(t)$$
$$y(t) = Cx(t) + Du(t) + e(t)$$

dkx=[d,k,x]中的任一元素为 1，则表示对应的参数矩阵 D、K 或 X0 作为估计参数；若为 0，则对应的参数矩阵固定为 0；

Maxsize,T，'trace'的定义参见函数 ar。

输出参数定义为：

th——以 Theta 模型格式表示的状态空间模型；

A0——选择作为极小化预测误差时域长度的辅助阶次 auxorder。

举例：对一个三输入二输出的对象进行基于子空间方法的状态空间模型辨识。该对象的输入输出关系为：

$$y_1(t) - 1.5y_1(t-1) + 0.4y_2(t-2) + 0.7y_1(t-2) = 0.2u_1(t-4) +$$
$$0.3u_1(t-5) + 0.4u_2(t) - 0.1u_2(t-1) + 0.15u_2(t-2) + e_1(t)$$
$$y_2(t) - 0.2y_1(t-1) - 0.7y_2(t-2) + 0.01y_1(t-2) = u_1(t) +$$
$$2u_2(t-4) + 3u_3(t-1) + 4u_3(t-2) + e_2(t)$$

图 1.4.8 所示为函数 n4sid 计算过程中绘制的不同阶次脉冲响应 Hankel 矩阵奇异值的比较图，并由用户通过命令窗口来选择模型的阶次：

```
A0 = eye(2);
A1 = [-1.5 0.4; -0.2 0];
A2 = [0.7 0 ; 0.01 -0.7];
B0 = [0 0.4 0;1 0 0];
B1 = [0 -0.1 0;0 0 3];
B2 = [0 0.15 0;0 0 4];
B3 = [0 0 0;0 0 0];
B4 = [0.2 0 0;0 2 0];
B5 = [0.3 0 0;0 0 0];
A = [A0 A1 A2];
B = [B0 B1 B2 B3 B4 B5];
th0 = arx2th(A,B,2,3);
e = randn(200,2);
u = [idinput(200),idinput(200),idinput(200)];
y = idsim([u e],th0);
z=[y u]
```

```
th=n4sid(z,[1:10],2)
```

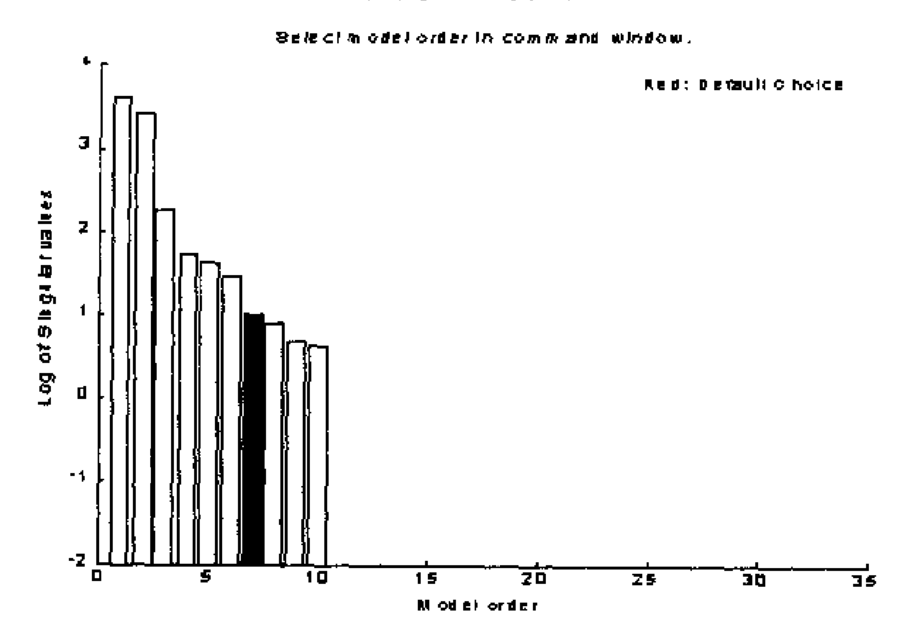


图 1.4.8　不同阶次模型的脉冲响应 Hankel 矩阵奇异值比较

1.4.6　一般线性输入输出模型辨识

一般的线性输入输出模型具有如下的形式：

$$A(q)y(t)=\frac{B_1(q)}{F_1(q)}u_1(t-nk_1)+\ldots+\frac{B_{nu}(q)}{F_{nu}(q)}u_{nu}(t-nk_{nu})+\frac{C(q)}{D(q)}e(t)$$

函数 pem 采用与函数 armax 相同的算法，即预测误差算法对上述一般的输入输出线性模型进行辨识，其使用说明如下。

1　pem

功能：采用预测误差算法辨识一般的线性输入输出模型。

格式：th = pem(z,nn)

th = pem(z,nn,'trace')

[th, iter_info] = pem(z,nn,index,maxiter,tol,lim,...maxsize,T,'trace')

说明：输入参数定义为：

z——对象的输入输出数据矩阵，z=[y u]。其中 y 为对象输出数据向量，u 为对象输入数据向量，y、u 均为列向量的形式；

nn——用于指定线性输入输出模型中各个多项式的阶次和纯时延大小，

nn=[na nb nc nd nf nk]

输出参数定义为：

th——以 Theta 模型格式表示的线性输入输出模型；

可选参数 iter_info，maxiter,tol,lim,maxsize,T,'trace'的定义参见函数 ar 有关辅

助变量 auxvar 的说明。

举例：仍然采用与函数 n4sid 示例中相同的对象输入输出数据，使用函数 pem 对该对象进行线性输入输出模型辨识。辨识得到的线性输入输出模型的频率响应波特图（输入 1 到输出 1），如图 1.4.9 所示。

```
A0 = eye(2);
A1 = [-1.5 0.4; -0.2 0];
A2 = [0.7 0 ; 0.01 -0.7];
B0 = [0 0.4 0;1 0 0];
B1 = [0 -0.1 0;0 0 3];
B2 = [0 0.15 0;0 0 4];
B3 = [0 0 0;0 0 0];
B4 = [0.2 0 0;0 2 0];
B5 = [0.3 0 0;0 0 0];
A = [A0 A1 A2];
B = [B0 B1 B2 B3 B4 B5];
th0 = arx2th(A,B,2,3);
e = randn(200,2);
u = [idinput(200),idinput(200),idinput(200)];
y = idsim([u e],th0);
z=[y u]
thc=canstart(z,5,3)
th=pem(z,thc)
thf=th2ff(th)
bodeplot(thf)
```

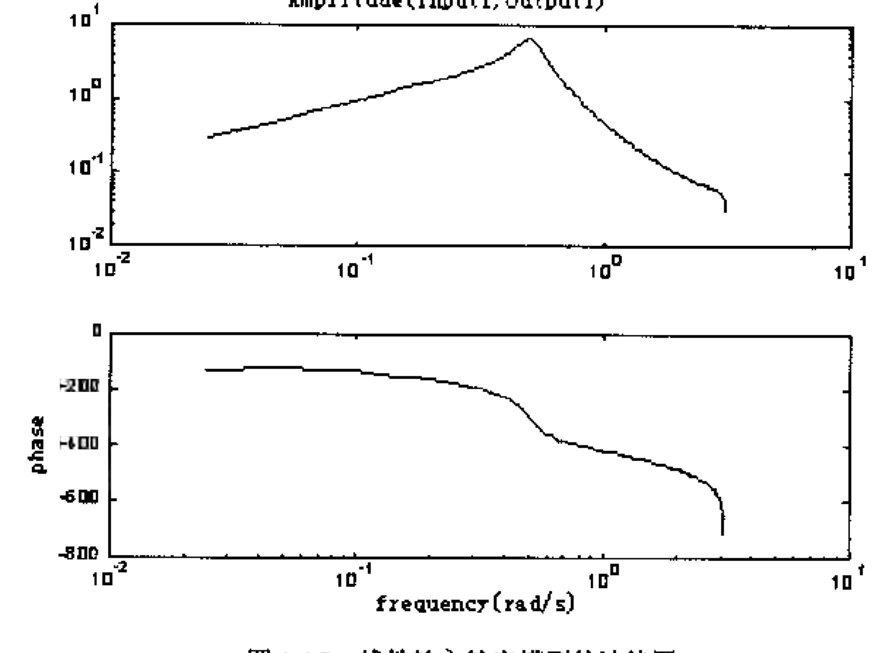


图 1.4.9 线性输入输出模型的波特图

1.5　递推参数模型辨识

上节介绍的参数模型辨识的有关函数，都是基于一次完成算法，而一次完成算法存在内存占用量大、难以用于在线辨识的缺点。因此，针对各种参数模型都提出了相应的递推辨识算法。递推辨识方法不仅减少了内存用量，而且能实现在线实时辨识。

在系统辨识工具箱中提供了各种参数模型的递推辨识函数，如表 1.5.1 所示。

表 1.5.1　参数模型的递推辨识函数

函数名称	功　能
rarmax	基于递推算法的 ARMAX 模型辨识
rarx	基于递推算法的 ARX 模型辨识
rbj	基于递推算法的 BJ 模型辨识
roe	基于递推算法的输出误差模型辨识
rpem	基于递推算法的一般线性输入输出模型辨识
rplr	基于递推伪线性回归算法的一般线性输入输出模型辨识

1.5.1　基于递推算法的 ARX、ARMAX 模型辨识

递推辨识算法通常具有如下的形式：

$$\hat{\theta}(t)=\hat{\theta}(t-1)+K(t)(y(t)-\hat{y}(t))$$

其中，$\hat{\theta}(t)$ 为时刻 t 的参数估值，$y(t)$ 为时刻 t 的测量输出，$\hat{y}(t)$ 为时刻 t 根据辨识模型得到的预测输出，$K(t)$为辨识增益矩阵。$K(t)$的选取通常为

$$K(t)=Q(t)\Phi(t)$$

对于 AR 和 ARX 等具有线性回归形式的模型，$\Phi(t)$ 为 $\hat{y}(t)$ 对于参数 θ 的变化梯度，即

$$y(t)=\Phi^T(t)\theta_0(t)+e(t)$$

其中，$\theta_0(t)$为对象的实际参数向量，$e(t)$为噪声。

如果假设对象的实际参数向量 $\theta_0(t)$的变化满足随机行走关系，即

$$\theta_0(t)=\theta_0(t-1)+w(t)$$

$$Ew(t)w^T(t)=R_1$$

则辨识算法的最优增益矩阵，可以根据 Kalman 滤波器计算得到。基于上述假设的完整递推辨识算法为：

$$\hat{\theta}(t)=\hat{\theta}(t-1)+K(t)(y(t)-\hat{y}(t))$$

$$\hat{y}(t)=\Phi^T(t)\hat{\theta}(t-1)$$

$$K(t)=Q(t)\Phi(t)$$

$$Q(t)=\frac{P(t-1)}{R_2+\Phi^T(t)P(t-1)\Phi(t)}$$

$$P(t)=P(t-1)+R_1-\frac{P(t-1)\Phi(t)\Phi^T(t)P(t-1)}{R_2+\Phi^T(t)P(t-1)\Phi(t)}$$

其中，R_2 为新息的方差矩阵。

上述基于 Kalman 滤波器的递推算法的一种改进方法是添加遗忘因子 λ。添加了遗忘因子后的递推算法公式修正为

$$Q(t)=\frac{P(t-1)}{\lambda+\Phi^T(t)P(t-1)\Phi(t)}$$

$$P(t)=\frac{1}{\lambda}P(t-1)-\frac{P(t-1)\Phi(t)\Phi^T(t)P(t-1)}{\lambda+\Phi^T(t)P(t-1)\Phi(t)}\frac{1}{\lambda}$$

另一种递推算法称为梯度算法，分为归一化梯度算法和未归一化梯度算法两种。其中归一化梯度算法的增益矩阵 $Q(t)=\dfrac{\gamma I}{|\Phi\ (t)|^2}$，而未归一化的梯度算法的增益矩阵 $Q(t)=\gamma I$，其中 I 为单位矩阵。

函数 rarx 和 rarmax 分别完成基于上述递推算法的 ARX 和 ARMAX 模型辨识，下面对这两个函数的使用方法进行介绍。

1　rarx

功能： 基于递推最小二乘方算法的 ARX 模型辨识。

格式： thm = rarx(z,nn,adm,adg)

[thm,yhat,P,phi] = rarx(z,nn,adm,adg,th0,P0,phi0)

说明： 输入参数定义为：

z——对象的输入输出数据向量，z=[y u]。其中 y 为对象的输出数据列向量，u 为对象的输入数据列向量，当 z=y 时，函数对 AR 模型进行辨识；

nn——指定 AR 或 ARX 模型的阶次。当进行 AR 模型辨识时，nn=na；当进行 ARX 模型辨识时，nn=[na nb nk]。其中 na、nb 分别为多项式 A(q)和 B(q)的阶次，nk 为对象的纯时延；

adm，adg——用于指定采用的递推最小二乘方算法的类型：

- adm='ff ',adg=lam——采用具有遗忘因子 $\lambda = lam$ 的递推最小二乘方算法，对应的输出参数 P 的定义为：$PR_2/2$ 近似等于估计参数的协方差矩阵，其中 R_2 为新息的方差；
- adm='ug',adg=gam——采用未归一化的梯度算法，且 γ=gam；
- adm='ng',adg=gam——采用归一化的梯度算法，且 γ=gam；
- adm='kf ',adg=R1——采用基于 Kalman 滤波器的递推算法，R_2=1，R_1=R1；

th0——指定模型参数的初始值，为行向量的形式；

P0——指定参数估计的协方差矩阵初值，缺省值为 $10^4 I$；

Phi0——数据向量的初值φ(0)，缺省值为零向量，数据向量φ(t)定义为

$$\varphi(t) = [y(t-1),...y(t-na),u(t-1),..u(t-nb-nk+1)]$$

输出参数定义为：

thm——参数估值矩阵，thm 的第 k 行为时刻 k 的参数估值，即

thm(k,:)=[a1,a2,...ana,b1,b2,...bnb]

yhat——输出的当前预测值矩阵；

P——当前参数估计的协方差矩阵；

Phi——当前的数据向量；

Psi——梯度向量 ψ。

当函数 rarx 用于在线计算时，参数 th0、phi0 和 P0 设定为上一步计算得到的输出参数 thm、phi 和 P。

举例：采用 AR 模型辨识的自适应噪声消除，设叠加了噪声的信号为

$$y(t) = 2*\sin(t) - 1.5*\sin(t-1) + e(t)$$

叠加了噪声的信号，如图 1.5.1 所示。

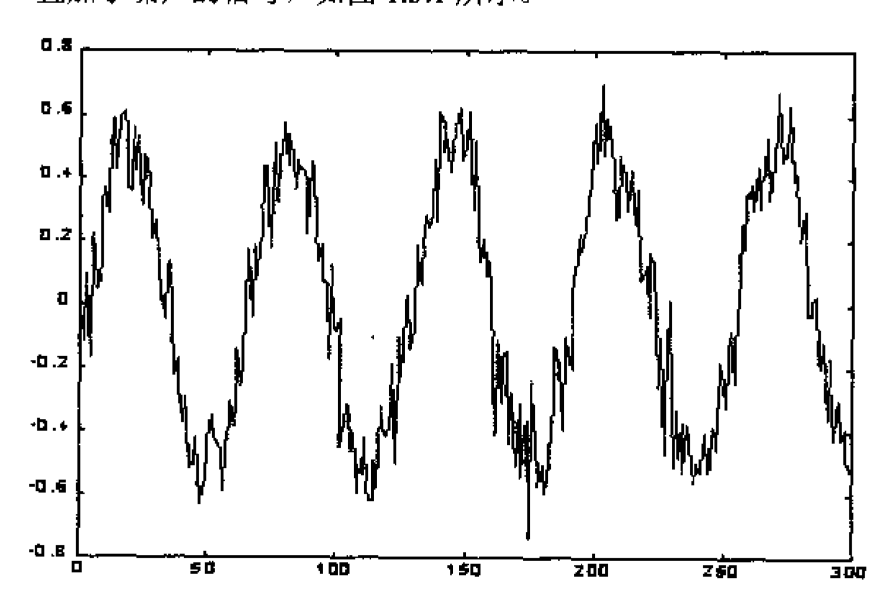


图 1.5.1　叠加了噪声的信号

下面的程序采用归一化的梯度算法进行 AR 模型辨识，并绘制噪声消除后的信号图形，如图 1.5.2 所示。

```
r=sin(0.1:0.1:31)
r1=r(1:300)
r2=r(2:301)
e=randn(1,300)
y=2.*r1-1.5.*r2+0.1.*e
z=[y',r1'];
[thm,yp]=rarx(z,[0 6 1],'ng',0.1);
plot(yp)
```

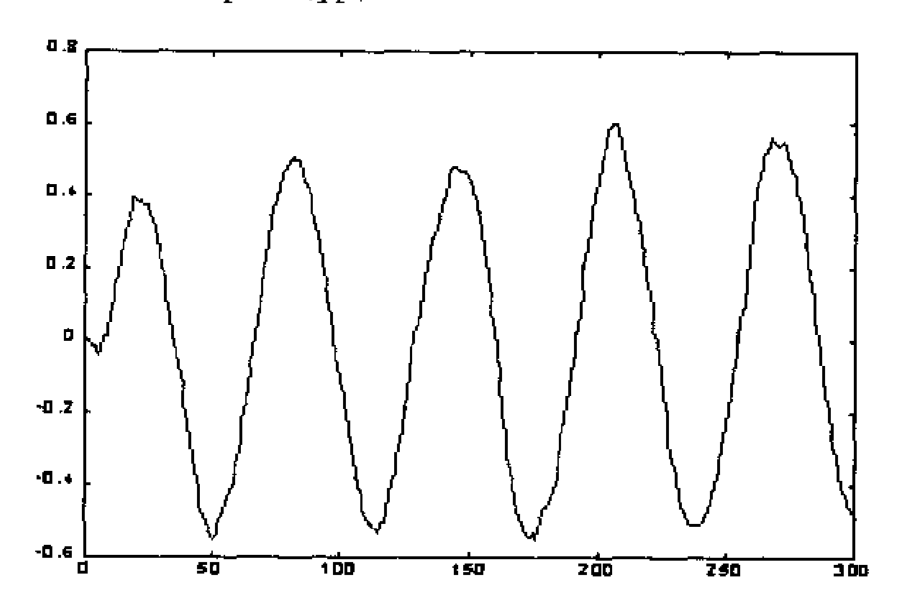


图 1.5.2 消除了噪声的预测输出

2 rarmax

功能： 采用递推算法进行 ARMAX 模型辨识。

格式： thm = rarmax(z,nn,adm,adg)

[thm,yhat,P,phi,psi] = rarmax(z,nn,adm,adg,th0,P0,phi0,psi0)

说明： 该函数只能用于单输入单输出对象的辨识。

输入参数定义为：

z——对象的输入输出数据向量，z=[y u]。其中 y 为对象的输出数据列向量，u 为对象的输入数据列向量；当 z=y 时，进行 ARMA 模型辨识；

nn——指定 ARMAX 模型的阶次，nn=[na nb nc nk]。其中 na、nb、nc 分别为多项式 A(q)、B(q)和 C(q)的阶次，nk 为对象的纯时延；当进行 ARMA 模型辨识时，nn=[na nc]；

adm，adg——用于指定采用的递推最小二乘方算法的类型，参见函数 rarx 的说明；

th0，P0，phi0，psi0 的定义与函数 rarx 相同。

输出参数定义为：

thm——参数估值矩阵，thm 的第 k 行为时刻 k 的参数估值，即

thm(k,:)=[a1,a2,…ana,b1,b2,…bnb]

yhat——输出的当前预测值矩阵；

P——当前参数估计的协方差矩阵；

Phi——当前的数据向量；

Psi——梯度向量 Φ。

举例：仍然考虑测量信号进行噪声消除的问题，设测量信号具有如下的形式：

$$y(t)=2*\sin(t)-1.5*\sin(t-1)+0.2*e(t)-0.1*e(t-1)$$

其中 e(t)为零均值白噪声信号，具有上述特性的测量信号如图 1.5.3 所示。采用 ARMAX 模型辨识方法进行噪声消除的程序如下：

```
r=sin(0.1:0.1:31)
r1=r(1:300)
r2=r(2:301)
e=randn(1,301)
e1=e(1:300)
e2=e(2:301)
y=2.*r1-1.5.*r2+0.2.*e1-0.1.*e2
z=[y',r1'];
[thm,yp]=rarmax(z,[0 6 2 1],'ng',0.1);
plot(yp)
```

进行了噪声消除处理后的信号如图 1.5.4 所示。

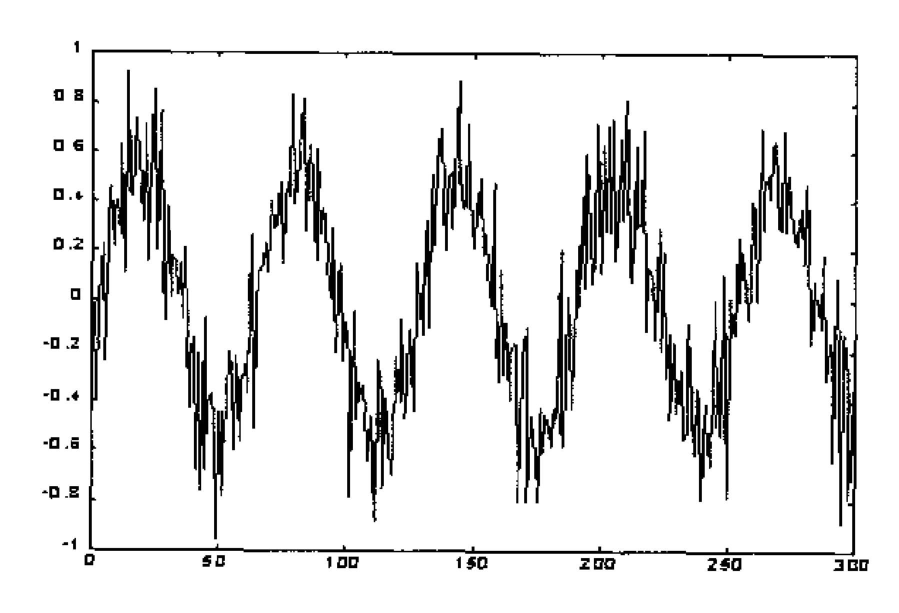


图 1.5.3　叠加了噪声的测量信号

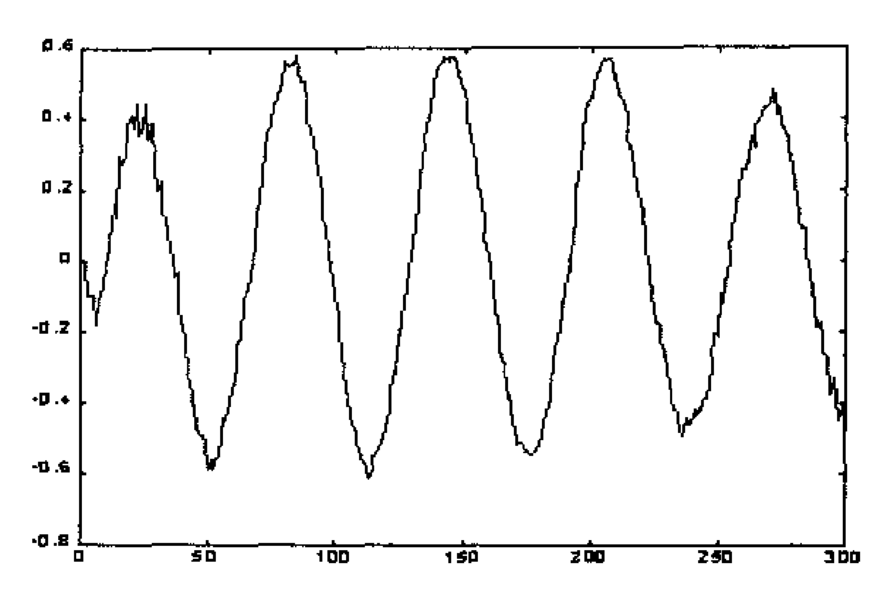


图 1.5.4 消除了噪声的信号

1.5.2 输出误差模型的递推辨识

函数 roe 用于完成输出误差模型的递推辨识，该函数采用的递推辨识算法与函数 rarx 和 rarmax 相同。

- roe

功能：输出误差模型的递推辨识。

格式：thm = roe(z,nn,adm,adg)

[thm,yhat,P,phi,psi] = roe(z,nn,adm,adg,th0,P0,phi0,psi0)

说明：输入参数定义为：

z——对象的输入输出数据向量，z=[y u]，其中 y 为对象的输出数据列向量， u 为对象的输入数据列向量；

nn——指定输出误差模型的阶次，nn=[nb nf nk]，其中 nb、nf 分别为输出误差模型多项式 B(q)和 F(q) 的阶次，nk 为对象的纯时延；

th0，P0，phi0，psi0 的定义与函数 rarx 相同；

adm，adg——用于指定采用的递推最小二乘方算法的类型，参见函数 rarx 的说明。

输出参数定义为：

thm——参数估值矩阵，thm 的第 k 行为时刻 k 的参数估值，即

thm(k,:)=[b1,b2,...bnb,f1,f2,...fnf]；

yhat——输出的当前预测值矩阵；

P—— 当前参数估计的协方差矩阵；

Phi——当前的数据向量；

Psi——梯度向量 Φ。

1.5.3　Box-Jenkins 模型的递推辨识

Box-Jenkins 模型具有下面的形式：

$$y(t)=\frac{B(q)}{F(q)}u(t-nk)+\frac{C(q)}{D(q)}e(t)$$

该模型的递推辨识可以利用函数 rbj。

- **rbj**

功能：Box-Jenkins 模型的递推辨识。

格式：thm = rbj(z,nn,adm,adg)

[thm,yhat,P,phi,psi] = rbj(z,nn,adm,adg,th0,P0,phi0,psi0)

说明：输入参数定义为：

z——对象的输入输出数据向量，z=[y u]，其中 y 为对象的输出数据列向量，u 为对象的输入数据列向量；

nn——指定 Box-Jenkins 模型的阶次，nn=[nb nc nd nf nk]，其中 nb、nc、nd 和 nf 分别为 Box-Jenkins 模型中多项式 B(q)、C(q)、D(q)和 F(q)的阶次，nk 为对象的纯时延；

adm，adg——用于指定采用的递推最小二乘方算法的类型，参见函数 rarx 的说明；

th0，P0，phi0，psi0 的定义与函数 rarx 相同。

输出参数的定义为：

thm——参数估值矩阵，thm 的第 k 行为时刻 k 的参数估值，即

thm(k,:)=[b1,b2,…bnb,c1,…cnc,d1,d2,…dnd,f1,f2,…fnf]

yhat——输出的当前预测值矩阵；

P——当前参数估计的协方差矩阵；

Phi——当前的数据向量；

Psi——梯度向量 Φ。

1.5.4　一般线性输入输出模型的递推辨识

一般线性输入输出模型的形式如下：

$$A(q)y(t)=\frac{B_1(q)}{F_1(q)}u_1(t-nk_1)+\ldots+\frac{B_{nu}(q)}{F_{nu}(q)}u_{nu}(t-nk_{nu})+\frac{C(q)}{D(q)}e(t)$$

对该模型的递推辨识可以采用两种方法，即递推预测误差方法和伪线性回归方法，对应的系统辨识工具箱函数分别为 rpem 和 rplr,下面对这两个函数进行介绍。

1 rpem

功能：基于递推预测方法的线性输入输出模型辨识。

格式：thm = rpem(z,nn,adm,adg)

[thm,yhat,P,phi,psi] = rpem(z,nn,adm,adg,th0,P0,phi0,psi0)

说明：输入参数中除 nn 外，其余参数的定义与函数 rarx、rarmax 等相同。nn 的定义为 nn=[na,nb,nc,nd,nf,nk]，其中 na、nb、nc、nd 和 nf 分别为模型中对应多项式的阶次，nk 为对象的纯时延，对于多输入系统，相应的阶次为行向量的形式；

输出参数中除 thm 外，其余参数与函数 rarx 相同，thm 的定义如下：

thm(k,:)=[a1,a2,...ana,b1,...bnb,c1,...cnc,d1,...dnd,f1,...fnf]

举例：多变量系统的一般线性输入输出模型递推辨识，设多变量系统的输入输出特性为

$$y_1(t)-1.5y_1(t-1)+0.7y_1(t-2)=0.2u_1(t-2)+0.3u_1(t-1)+0.4u_2(t-1)-0.1u_2(t-2)+0.15\,u_3(t-2)+e_1(t)$$

采用下面的程序进行线性输入输出模型的递推辨识，得到的预测输出和实际输出曲线，如图 1.5.5 所示（图中实线为实际输出，虚线为预测输出）。

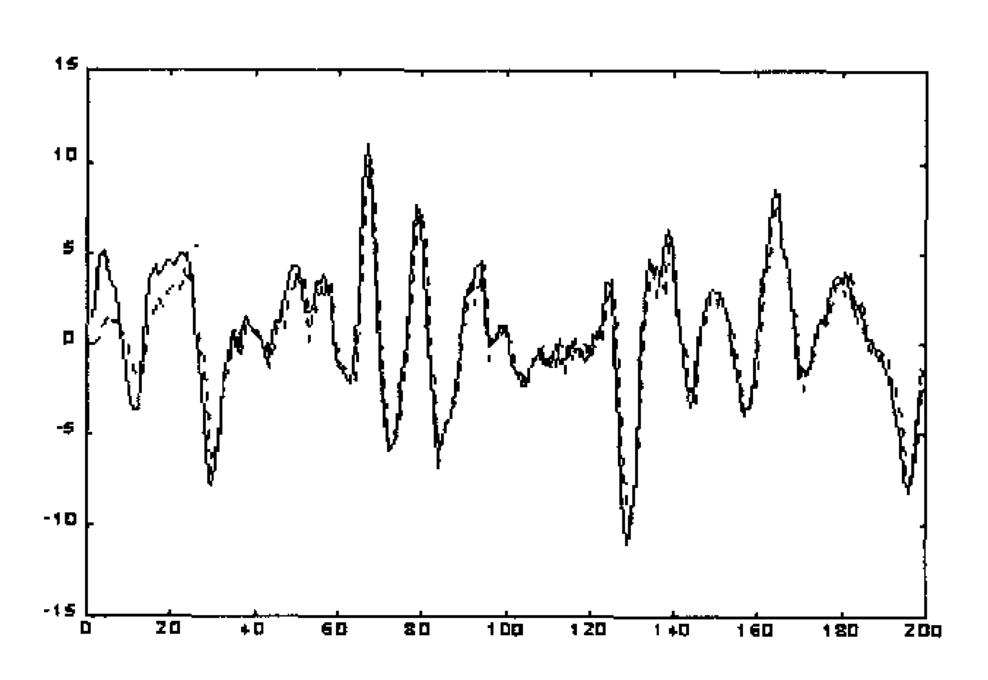


图 1.5.5 预测输出与实际输出比较

```
A0 = [1 -1.5 0.7];
B0 = [0 0.4 0];
B1 = [0.3 -0.1 0];
B2 = [0.2  0 0.15 ];
A = A0 ;
B = [B0 B1 B2 ];
```

```
th0 = arx2th(A,B,1,3);
e = randn(200,1);
u = [idinput(200),idinput(200),idinput(200)];
y = idsim([u e],th0);
z=[y u]
nb=[2 2 1]
nf=[1 1 1]
[thm,yhat] = rpem(z,[1 nb 1 1 nf 1 1 2] ,'ng',0.1)
plot(y,'-')
hold on
plot(yhat,':')
```

2 rplr

功能： 基于伪线性回归的一般线性输入输出模型辨识。

格式： thm = rplr(z,nn,adm,adg)

[thm,yhat,P,phi] = rplr(z,nn,adm,adg,th0,P0,phi0)

说明： 该函数的输入输出参数定义与函数 rpem 相同，且都是对一般的输入输出线性模型进行辨识，但函数 lplr 使用了另一种梯度算法即递推伪线性回归方法。另外，函数 lplr 仅能用于单输入输出系统。

当用于 ARMAX 模型辨识时，nn=[na nb nc 0 0 nk]；当用于输出误差模型辨识时，nn=[0 nb 0 0 nf nk]。

举例： 考虑如下的单输入输出对象

$$y(t)-1.5y(t-1)+0.7\,y(t-2)=0.2u(t-2)+0.3\,u(t-1)+0.5\,u(t-3)+e(t)$$

采用 lplr 进行对象的线性输入输出模型辨识，得到的预测输出与实际输出的比较曲线，如图 1.5.6 所示。

```
A0 = [1 -1.5 0.7];
A = A0 ;
B = [0 0.3 0.2 0.5];
th0 = arx2th(A,B,1,1);
e = randn(200,1);
u = idinput(200);
y = idsim([u e],th0);
z=[y u]
nb=2
nf=1
[thm,yhat] = rplr(z,[1 nb 1 1 nf 1 ] ,'ng',0.1)
plot(y,'-')
hold on
plot(yhat,':')
```

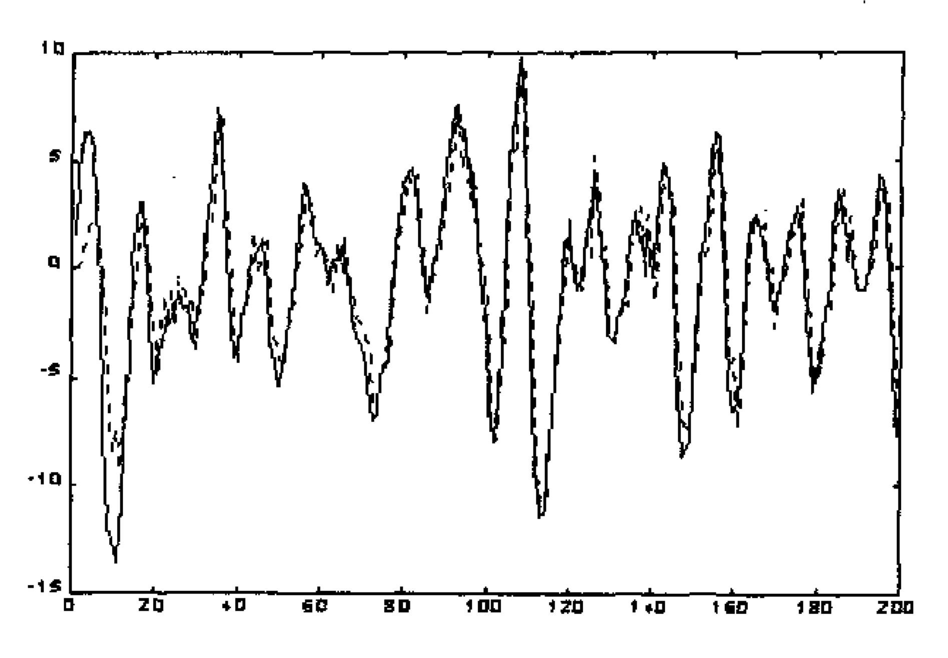


图 1.5.6 预测输出与实际输出曲线

1.5.5 基于数据分段的估计和辨识

实际工程应用中，可能由于某种原因如出现故障而导致系统特性的突然改变。对这些条件下获得的系统输入输出数据进行模型辨识，需要能够自动判断系统特性发生改变的时刻，并对数据进行分段估计和辨识。

系统辨识工具箱的函数 segment 能够完成基于数据分段的估计和辨识。该函数同时采用多个递推 Kalman 滤波器对系统的 ARX 或 ARMAX 模型进行辨识和估计，每个 Kalman 滤波器对应一种数据分段假设，这些 Kalman 滤波器得到的模型参数估计不断被加以评价，具有较小可能性概率（称为后验概率）的模型参数将被新的参数代替。函数 segment 计算结束后得到的模型，为具有最大后验概率的模型参数和相应的分段时刻。

- **segment**

功能：基于数据分段的 ARX、ARMAX 模型辨识。

格式：segm = segment(z,nn)

[segm,V,thm,R2e] = segment(z,nn,R2,q,R1,M,th0,P0,ll,mu)

说明：输入参数定义为：

z——对象的输入输出数据矩阵，z=[y u]。其中 y 为对象输出数据向量，u 为对象输入数据向量，y、u 均为列向量的形式；若 z=y，则进行时间序列的 AR 或 ARMA 模型辨识；

nn——用于指定 ARMAX 模型的阶次和纯时延大小，nn=[na nb nc nk]。其中 na 为 AR 部分的阶次，nb 为 MA 部分的阶次，nc 为噪声特性的阶次，nk 为对象的纯时延；对于 ARX 模型辨识,nn=[na nb nk]；对于 AR 模型辨识，nn=na；对于 ARMA 模型辨识，na=[na nc]；

R2——模型的息方差，缺省值为息方差的估值；
R1——模型参数发生突变时的参数协方差矩阵；
q——模型在任意时刻发生突变的概率，缺省值为 0.01；
M——指定同时采用的估计模型个数，缺省值为 5；
Th0——指定参数的初始值，缺省值为 0；
P0——初始的参数方差矩阵，缺省值为 10I；
ll——指定每个估计模型的最短有效时间，即在分段估计过程中的每个估计模型至少经过 ll 个时间步长才可能被其他模型代替，缺省值为 1；
mu——估计新息方差的遗忘因子。

输出参数定义为：

segm——基于分段估计的模型参数矩阵，该矩阵的第 k 行对应第 k 时刻具有最大后验概率的参数估计；
thm——没有进行分段估计的模型参数矩阵；
V——分段参数模型的输出预测误差平方和；
R2e——新息方差的估值向量，第 k 个元素对应 k 时刻的估值。

举例： 下面演示函数 segment 对给定数据的分段结果。

设对象的标称输入输出特性为

$$y(t)-c$$

当对象输入输出特性变为 $y(t)=\sin(t)/3$ 时，采用函数 segment 对给定的标称模型进行分段辨识，其分段辨识结果如图 1.5.7 所示。

```
y = sin([1:50]/3)';
thm = segment([y,ones(size(y))],[0 1 1],0.1);
plot([thm,y])
```

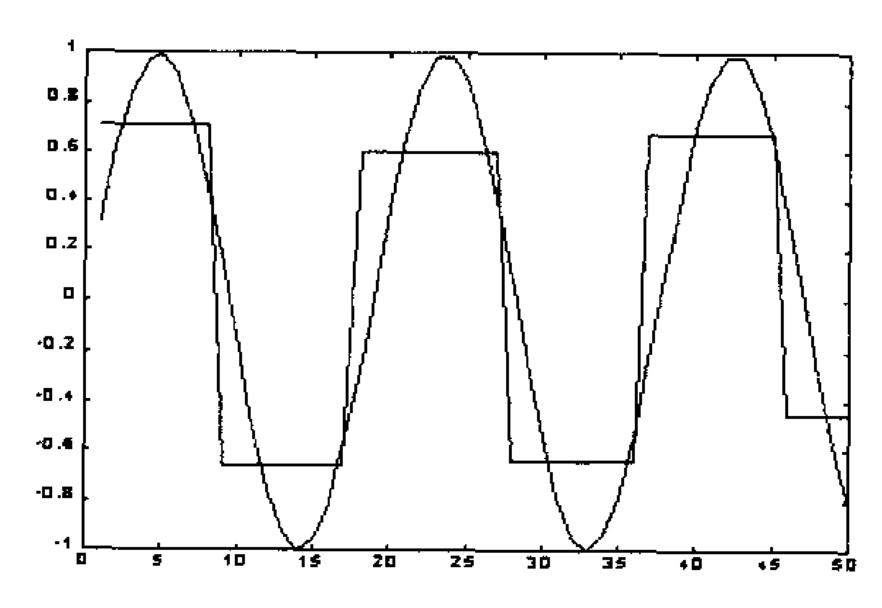


图 1.5.7　数据分段辨识结果

1.6 模型验证与仿真函数

模型验证在系统辨识中是不可缺少的步骤，在系统辨识工具箱中提供的模型验证函数如表 1.6.1 所示。

表 1.6.1 模型验证与仿真函数

函数名称	功　能
compare	将模型的预测输出与对象实际输出进行比较
idsim	进行模型仿真计算
pe	计算预测误差
predict	预测未来输出
idinput	生成输入信号
resid	计算和检验模型的残差

1.6.1 模型仿真函数

为进行模型辨识和仿真，往往需要生成不同特性的输入信号，如高斯随机信号、二值随机信号等。函数 idinput 用于完成上述功能。

在进行模型验证时，需要对辨识模型进行仿真计算，函数 idsim 用于对 Theta 格式的模型进行仿真。

1 idinput

功能：生成不同类型的辨识输入信号。

格式：u = idinput(N)

u = idinput(N,type,band,levels)

u = idinput(N,'sine',band,levels,auxvar)

说明：输入参数定义为：

N——生成的信号数据长度；

type——生成的信号类型，包括如下几种：

type='rs'——高斯随机信号；

type='rbs'——二值随机信号；

type='prbs'——二值伪随机信号；

type='sine'——二值随机信号；

缺省值为 type='rbs'；

band——指定信号的带宽，当信号类型为'rb'、'rbs'或'sine'时，band 为两个元素的行向量形式，即 band=[wlow whigh]，其中 wlow 和 whigh 分别为信号频率带宽的下界和上界，band 的缺省值为[0 1]，即生成白噪声信号；

当信号类型为'prbs'时，band=[twologp M]，其中 twologp 决定了伪随机二值信号的周期 T=2^twologp-1，M 指定在 1/M 长度的时间区间内为常数，当 twologp=0 或 twologp=18 时，生成的信号周期得到最大；

levels —— 用于决定输入信号幅值的上下界， 定义为行向量的形式，即 levels=[minu maxu]；当信号类型为'rs'时，minu 为高斯信号的均值减 1，maxu 为高斯信号的均值加 1；

auxvar——用于输入信号类型为 'sine'时的辅助变量，其格式定义为

auxvar = [no_of_sinusoids, no_of_trials]

其中，no_of_sinusoids 用于决定生成信号的正弦函数个数，no_of_trials 用于决定在进行正弦函数相位随机化时使信号幅值极小的计算次数；缺省值为 auxvar=[10 10]。

输出参数 u 为生成的信号数据向量。

举例：%生成零均值高斯随机信号，如图 1.6.1 所示。

```
u=idinput(300,'rs',[0 1],[-1 1])
Plot(u)
```

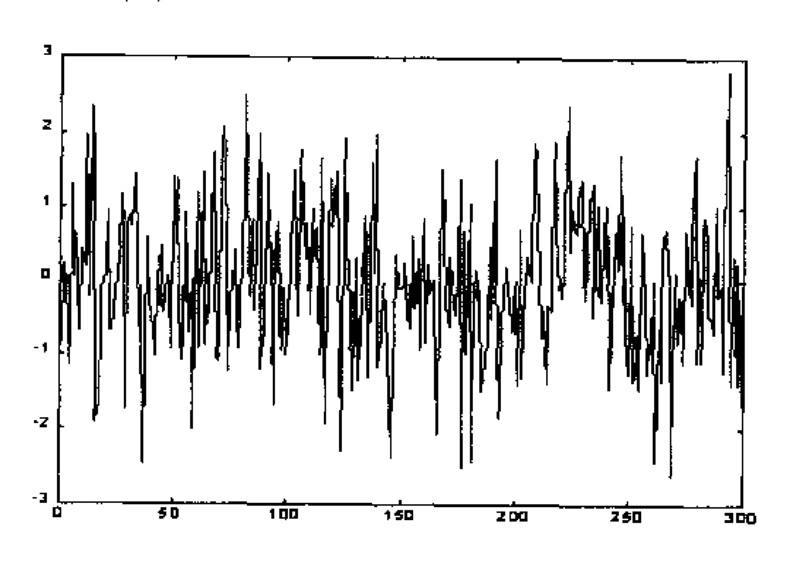


图 1.6.1　零均值高斯随机信号

2　idsim

功能：Theta 格式模型的仿真。

格式：y = idsim([u e],th)

[y, ysd] = idsim(u,th)

说明：输入参数定义为：

[u e]——指定输入和噪声数据向量；当省略噪声数据向量 e 时，进行无噪声的模型仿真；

th——对象的 Theta 格式模型。

输出参数定义为：

y——仿真输出数据向量；

ysd——仿真输出的标准差。

在仿真计算中，噪声序列 e 被尺度化（尺度化因子为λ，λ由 th 中的噪声方差决定；为得到正确的噪声影响相关，可选择零均值、方差为单位矩阵的噪声信号。

举例：对如下的 ARX 模型进行仿真，仿真输出曲线如图 1.6.2 所示。

$$2y(t)-0.5y(t-1)+0.2y(t-2)=u(t)+0.3u(t-1)+0.2u(t-2)+e(t)$$

```
A= [2 -0.5 0.2];   B = [1 0.3 0.2 ];
th0 = arx2th(A,B,1,1);
e = randn(200,1);
u = [idinput(200)];
y = idsim([u e],th0);
plot(y)
```

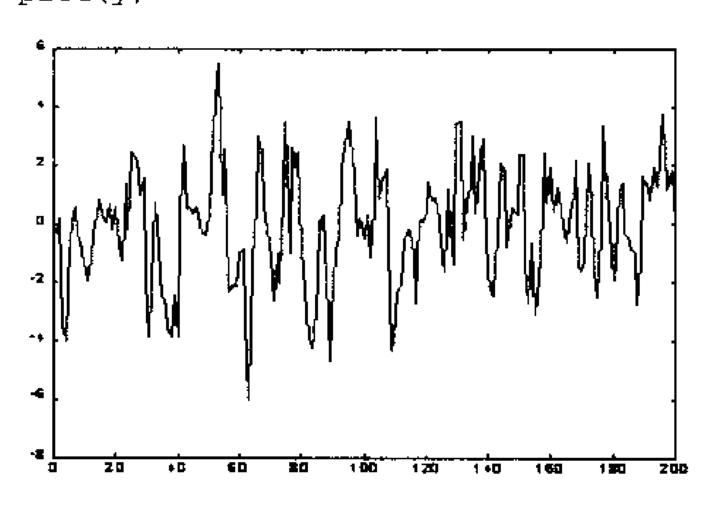


图 1.6.2 ARX 模型的仿真输入

1.6.2 模型预测输出和预测误差的计算

在获得对象的辨识模型后，检验该模型有效性的一种方法就是计算辨识模型利用实际的输入输出历史数据预测未来输出的精度。函数 predict 用于计算辨识模型的预测输出。

1 predict

功能：根据历史输入输出数据计算辨识模型的预测输出。

格式：yp = predict(z,th)

[yp,thpred] = predict(z,th,k)

说明：输入参数定义为：

z——对象的实际输入输出历史数据，z=[y u]；其中矩阵 y 的第 r 列对应第 r 个输出的数据；

th——以 Theta 格式表示的对象辨识模型；

k——预测的时间长度。

输出参数定义为：

yp——辨识模型的预测输出；

thpred——当辨识模型为输入输出模型时对应的用于 k 步预测计算的模型。

实际输出与预测输出曲线，分别如图 1.6.3 和图 1.6.4 所示。

举例：

```
th0 = poly2th([1 -0.99],[],[1 -1 0.2]);
y = idsim(randn(400,1),th0);
th = armax(y(1:200),[1 2]);yp = predict(y,th,4);
plot(y(201:400)) ;plot(yp(201:400))
```

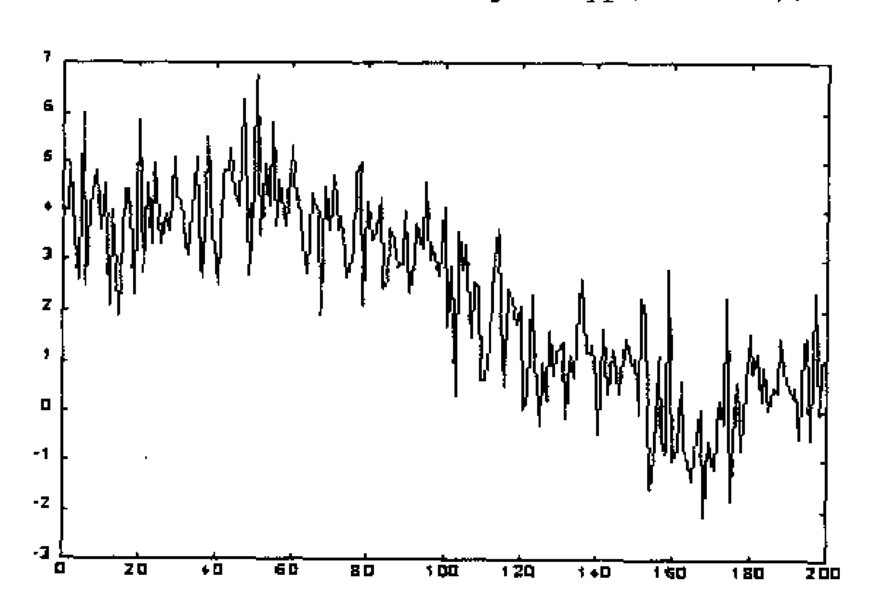


图 1.6.3 对象实际输出曲线

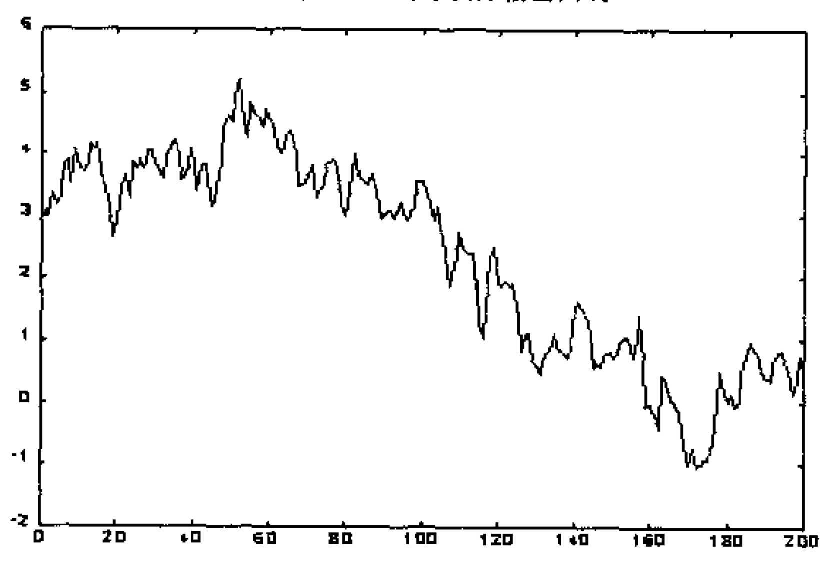


图 1.6.4 预测输出曲线

系统辨识工具箱中，能够用于计算辨识模型预测输出的另一个函数为 compare，该函数在计算模型输出的基础上对实际输出和模型输出进行比较。

2 compare

功能：计算模型输出并与实际输出比较。

格式：compare(z,th)

[yh,fit] = compare(z,th,k,sampnr,leveladj)

说明：输入参数定义为：

z——对象的实际输入输出历史数据，z=[y u]；其中矩阵 y 的第 r 列对应第 r 个输出的数据；

th——以 Theta 格式表示的对象辨识模型；

k——预测的时间长度，k=inf 时则不进行预测计算；

sampnr——指定用于计算模型适应度 fit 和绘图的数据点向量；

leveladj——指定是否对实际输出和模型输出进行幅值调整，当 leveladj='yes' 时，实际输出和模型输出均以 0 作为初始值。

输出参数定义为：

yh——模型输出向量或矩阵；

fit——模型的适应度值，其计算公式为

$$fit = \|yh - y\| / \sqrt{N}$$

其中 N 为输出数据的长度。

举例：下面的例子将对象的输入输出数据分为两部分，一部分用于 ARMAX 模型辨识，另一部分进行输出预测以验证模型的有效性。

```
v=randn(501,1)
v1=v(2:501)
v2=0.2.*v(1:500)
A - [1 -0.1 0.2];
B = [0 1 0.5];
th0 = poly2th(A,B);
u = idinput(500,'rbs');
y = idsim([u,v2,v1],th0);
z = [y,u];
ze = z(1:250,:);
zv = z(251:500,:);
th=armax(ze,[2 2 2 1]);
[yh fit]=compare(zv,th,6);
plot(yh-y(251:500))
```

计算得到的模型适应度 fit=0.21314，预测输出和预测误差的曲线分别如图 1.6.5 和图 1.6.6 所示。

函数 pe 和 resid 分别用于计算预测误差和在计算预测误差的基础上进行残差相关分析。

3　pe

功能：计算模型预测误差。

格式：e = pe(z,th)

说明：输入参数定义为：

z——输入输出数据向量或矩阵：z=[y u]。其中矩阵 y 的第 r 列对应第 r 个输出的数据；

th——以 Theta 格式表示的对象辨识模型。

输出参数定义为：

e——模型预测误差。

举例：

```
v-randn(301,1) ;
v1=v(2:301)
v2=0.2.*v(1:300)
A = [1 -0.1 0.2];
B = [0 1 0.5];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,v2,v1],th0);
z = [y,u];
th=armax(z,[2 2 2 1]);
e=pe(z,th)
```

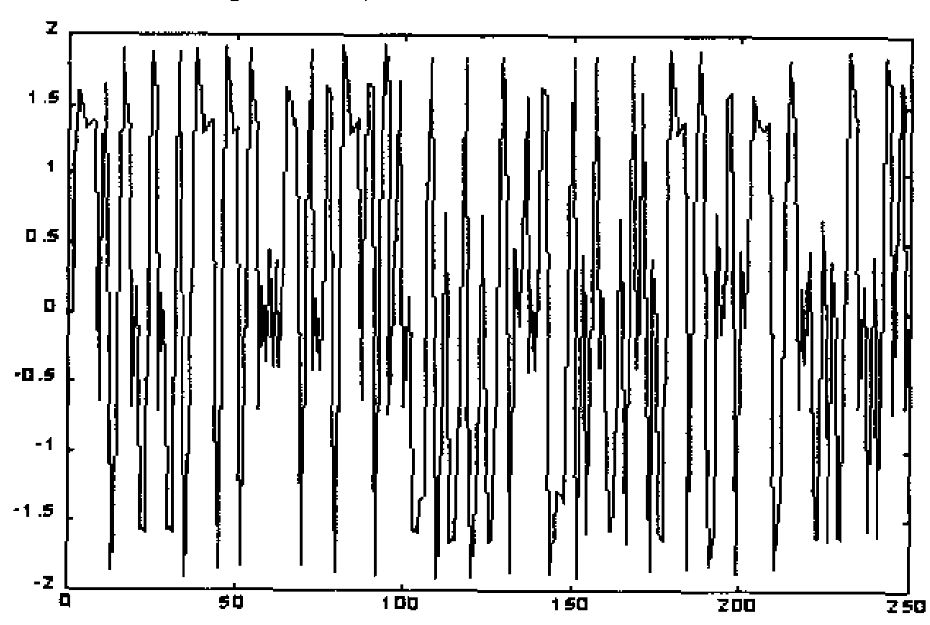


图 1.6.5　预测输出曲线

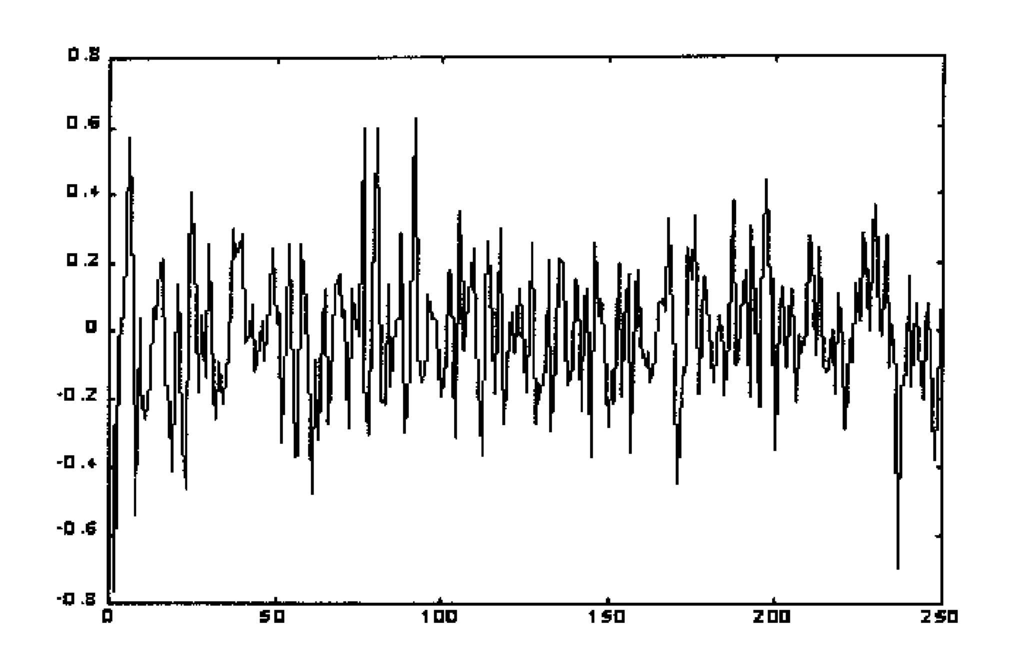


图 1.6.6 预测误差曲线

4 resid

功能： 计算模型预测误差并进行相关分析。

格式： [e,r] = resid(z,th)

[e,r] = resid(z,th,M,maxsize)

resid(r)

说明： 输入参数定义为：

z——输入输出数据向量或矩阵；z=[y u]；

th——以 Theta 格式表示的对象辨识模型；

M——指定相关函数的最大时间长度，缺省值为 25；

maxsize——参见函数 ar 关于辅助变量的说明。

输出参数定义为：

e——模型预测误差；

r——预测误差的自相关函数以及预测误差与输入的互相关函数；

当以 r 为输入参数时，函数 resid 将绘制相应的自相关函数和互相关函数曲线。

举例： 下面的程序在进行对象的 ARMAX 建模后，调用函数 resid 进行预测误差计算和相关分析，并绘制残差自相关函数和输入与残差的互相关函数曲线，如图 1.6.7 所示。

```
v=randn(301,1)
v1=v(2:301)
v2=0.2.*v(1:300)
A = [1 -0.1 0.2];
B = [0 1 0.5];
```

```
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,v2,v1],th0);
z = [y,u];
th=armax(z,[2 2 2 1]);
[e r]=resid(z,th)
resid(r)
```

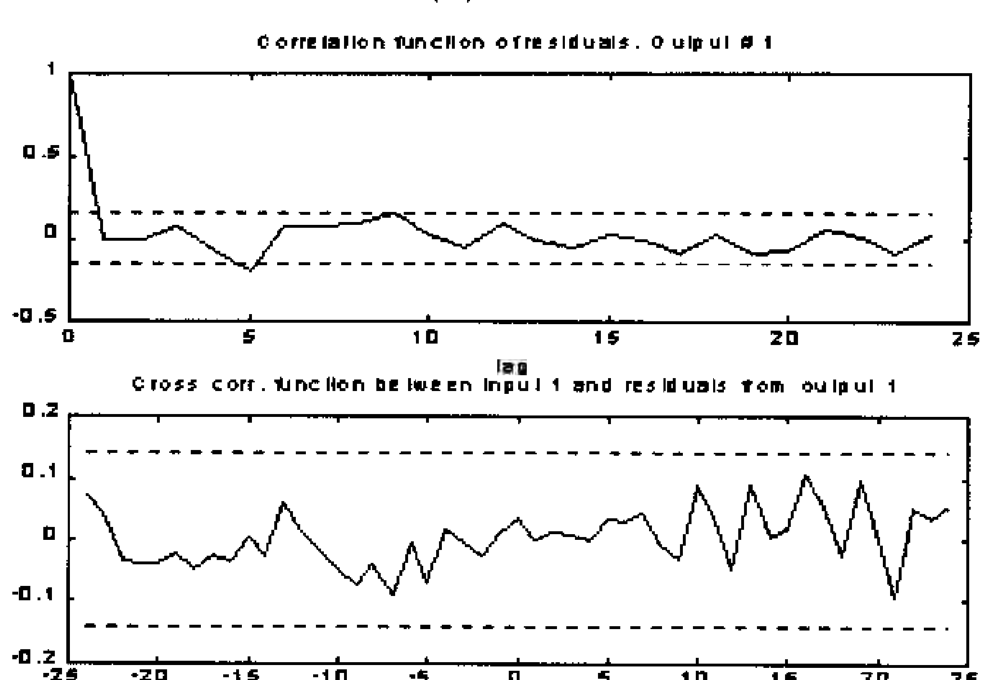


图 1.6.7　残差自相关函数（上图）与输入和残差的互相关函数曲线

1.7　系统辨识工具箱的交互式图形界面

在系统辨识工具箱中，除了提供各种以 Matlab 命令形式调用的功能函数外，还提供了一个交互式的图形界面工具。该图形界面工具能够方便地实现数据的预处理、模型类型的选择和参数估计以及模型验证和比较等功能。为启动系统辨识工具箱的图形界面，只需要在 Matlab 命令窗口键入命令 ident 即可。该图形界面的初始化窗口，如图 1.7.1 所示。

在图 1.7.1 的窗口中，菜单部分的功能主要包括打开或存储一个系统辨识对话进程(Session)以及相应的选项设定功能。菜单外的其余部分可以按功能分为三块，即左侧的数据视图、右侧的模型视图和中间的操作选择部分。本节将按上述三个部分的划分，介绍系统辨识工具箱图形界面的功能和用法。

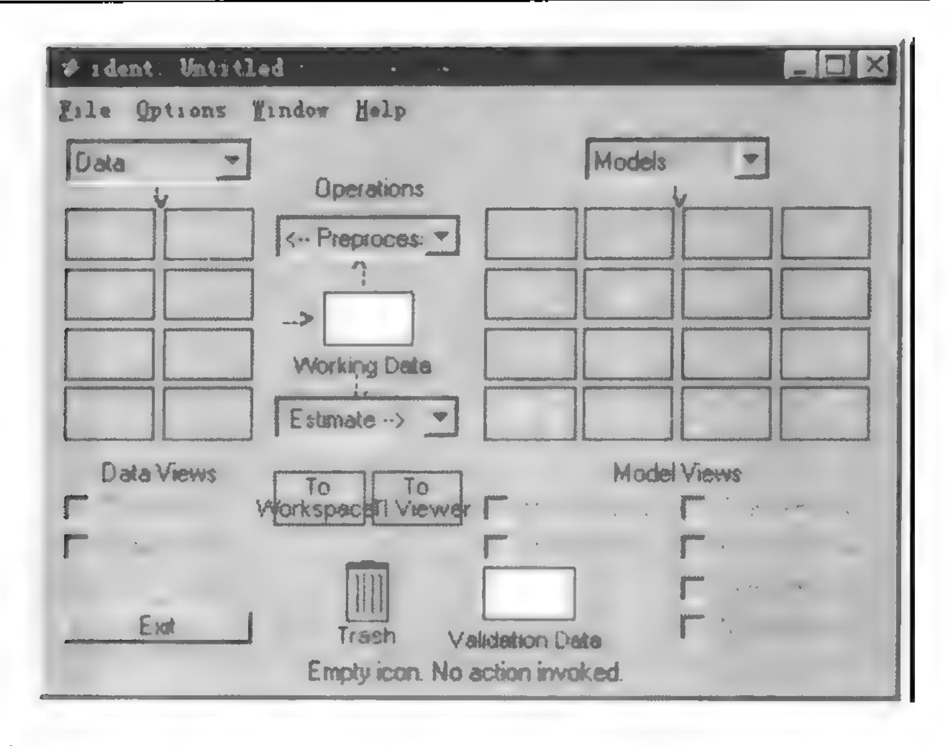


图 1.7.1 系统辨识工具箱的图形界面

1.7.1 数据视图

在图 1.7.1 中窗口的左侧为数据视图（Data Views）部分，包括输入输出数据的导入、数据变化曲线和频谱的绘制等功能。对象输入输出数据的导入可以在窗口左上方的下拉列表框中选择 Import，即可通过相应对话框的选择从 Matlab 工作空间（Workspace）中导入输入输出数据。导入输入输出数据的对话框，如图 1.7.2 所示。

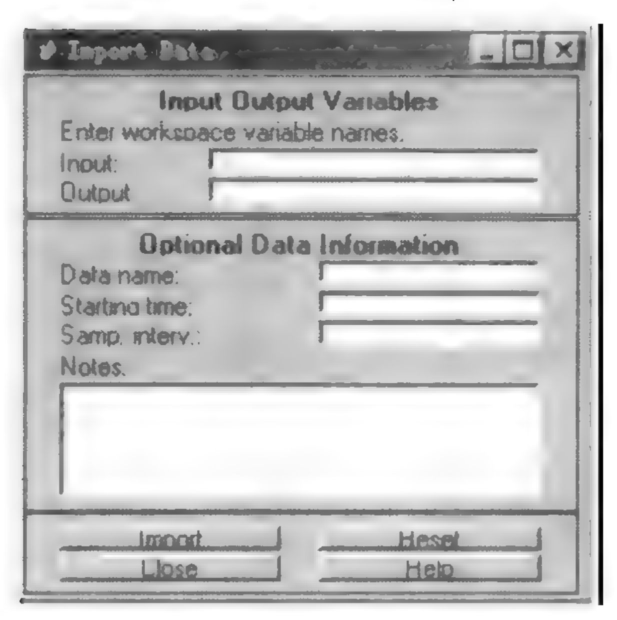


图 1.7.2 导入数据的对话框

在图 1.7.2 的对话框中，通过指定 Matlab 工作空间中对应的变量名称，即可导入输入输出数据。

在导入输入输出数据之后，可以对这些数据进行预处理，如消除趋势项等。这些预处理可以在图形界面的操作选择部分进行。

下面以一个系统辨识实例来说明图形界面的功能和用法，这个实例的有关数据存储在 Matlab 的 SID 文件'dryer.sid'中，可以直接从图形界面的文件菜单中选择打开。打开该文件后，将自动导入原始输入输出数据以及经过预处理后的三组输入输出数据。此时，图形界面的显示如图 1.7.3 所示。

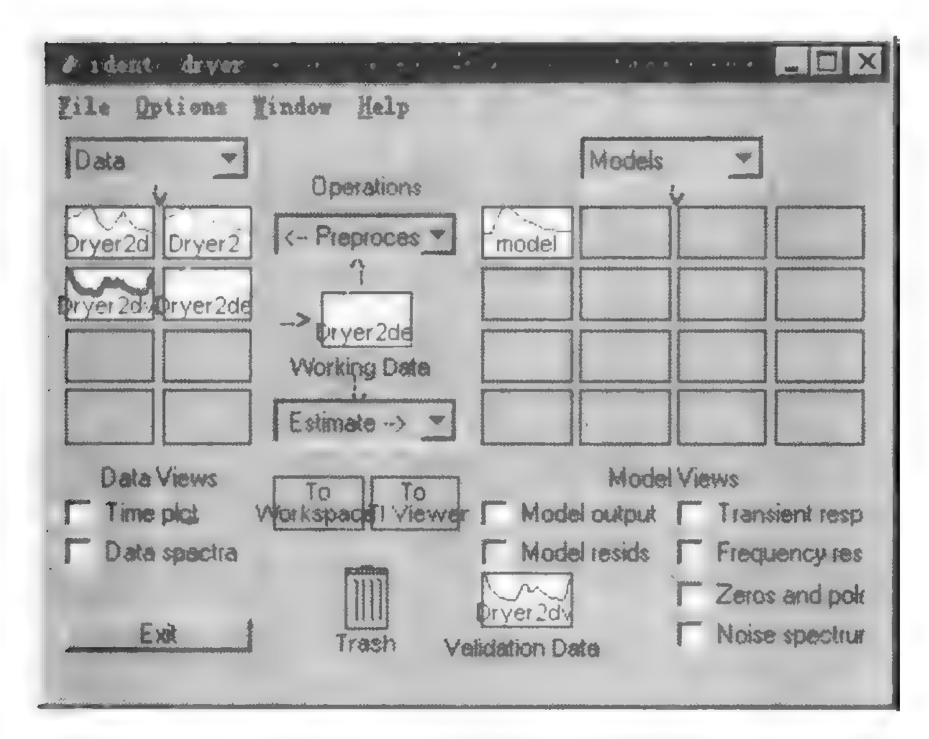


图 1.7.3　打开 SID 文件 dryer 后的图形界面

在图 1.7.3 中，数据视图部分包括四组输入输出数据，其中 Dryer2 为对象的原始输入输出数据，Dryer2d 为消除了趋势项的输入输出数据，Dryer2de 和 Dryer2dv 分别用于模型辨识和模型验证。Dryer2de 由 Dryer2d 的前 500 个数据点构成，Dryer2dv 由 Dryer2d 的后 500 个数据点构成。Dryer2d、Dryer2de 和 Dryer2dv 可以通过在图形界面的操作选择部分进行预处理完成，也可以在 Matlab 工作空间中利用有关命令行函数完成。实现上述预处理的 Matlab 命令为：

```
load dryer2
Dryer2=[[y2],[ u2]];
Dryer2d = dtrend(Dryer2,0)
Dryer2de = Dryer2d([1:500],:)
Dryer2dv = Dryer2d([501:1000],:)
```

其中函数 dtrend 用于消除输入输出数据中趋势项，即所有数据点减去均值。

在数据视图部分的下部有两个检查框，分别用于指定绘制数据的时间变化曲线和频谱

曲线。对象原始输入输出数据的时间变化曲线和频谱曲线，如图 1.7.4 和 1.7.5 所示。

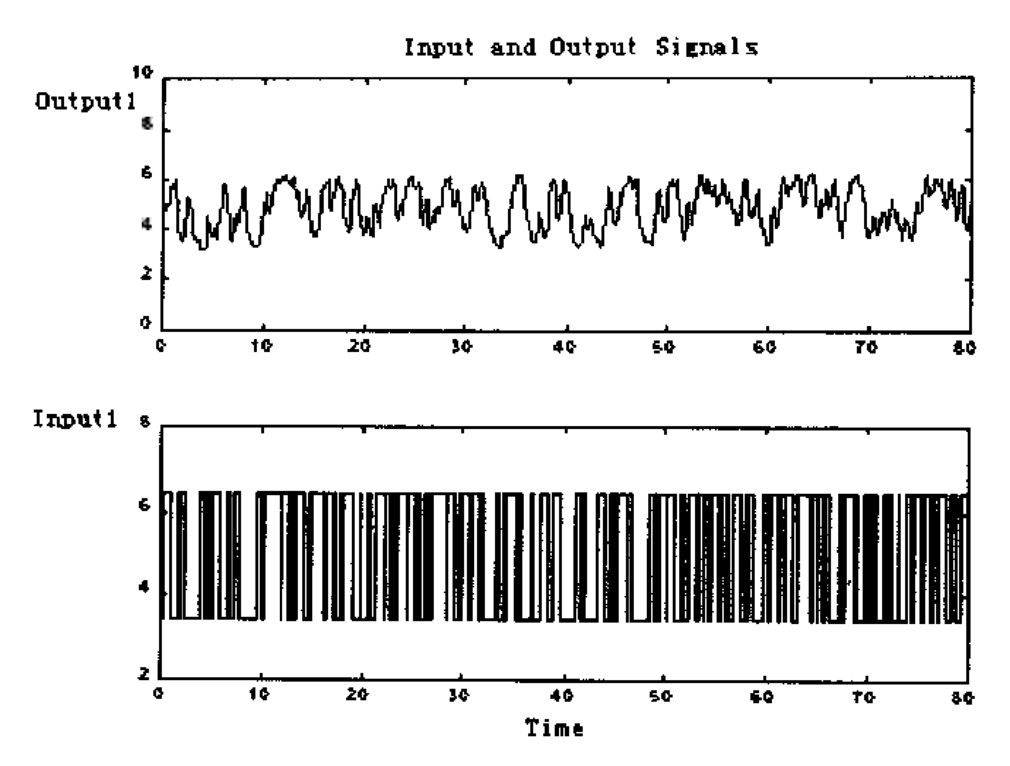


图 1.7.4　对象原始输入输出数据的时间变化曲线

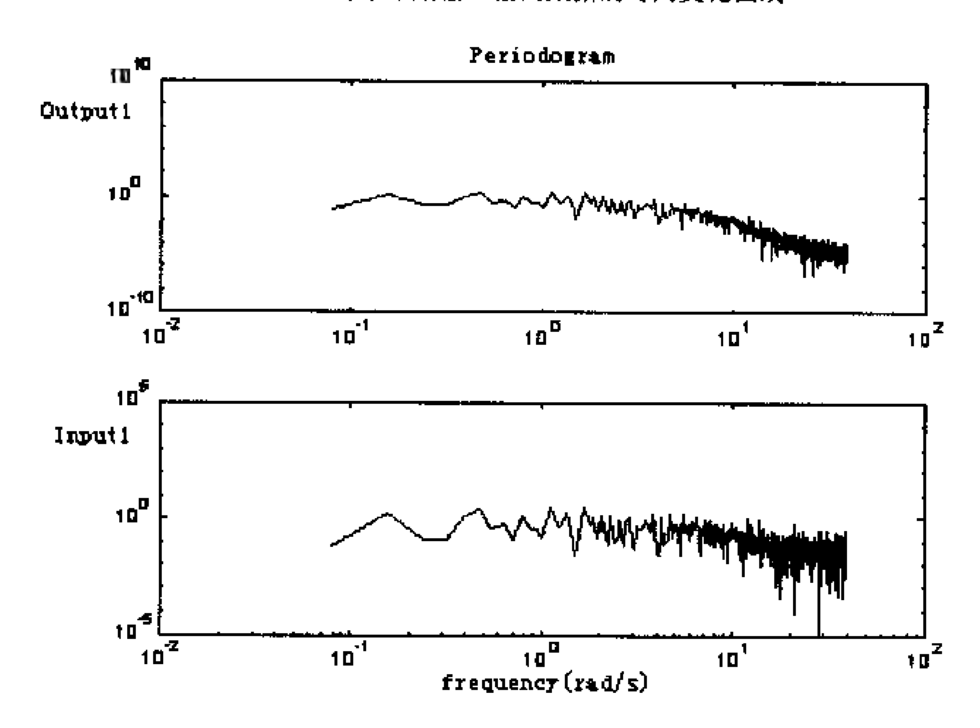


图 1.7.5　对象原始输入输出数据的频谱曲线

1.7.2 操作选择

在图 1.7.1 所示的图形界面的中部为操作选择部分，包括两个下拉列表框。上部的下拉列表框用于进行有关数据操作的选择，下部的下拉列表框用于进行模型类型的选择。

在上部的数据操作选择列表框中，可以对输入输出数据进行有关预处理，包括消除趋势项、滤波等。输入输出数据在经过消除趋势项操作后的时间变化曲线，如图 1.7.6 所示。

在下部的列表框中，可以选择模型的类型并通过相应的对话框输入阶次等信息。在两个列表框的中间有一个区域用于指示当前的工作数据，可以通过利用鼠标将数据视图部分的有关数据图标拖曳至该区域，来指定当前的工作数据。

在操作选择部分还包括两个选项：To Workspace 和 To Viewer，分别用于将图形界面的有关计算结果输出到 Matlab 工作空间和 LTI Viewer。

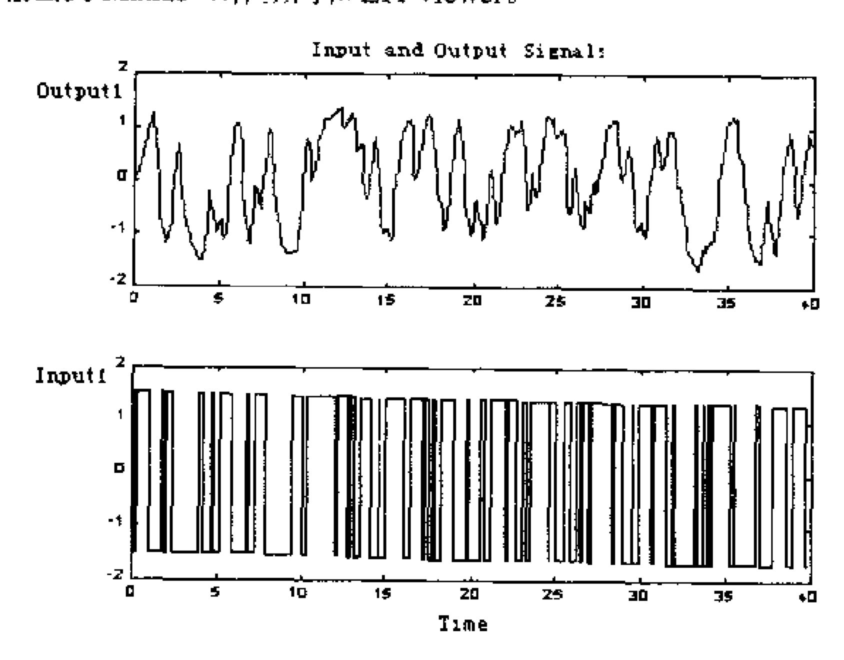


图 1.7.6　消除了趋势项的输入输出数据

1.7.3 模型视图

在图 1.7.1 的右侧部分为图形界面的模型视图，其主要功能包括：不同类型模型的选择和切换、模型的验证和特性曲线绘制。在模型视图的上部的下拉列表框，用于进行模型类型的选择。当选择某一种模型类型并输入相应的信息后，即可将该模型对应的图标加入列表框下部的模型图标区域。在模型图标区域显示当前已加入的模型类型的图标，通过鼠标可对这些模型进行选择。

在模型图标区域的下方有多个检查框，可用于选择对应的功能，这些功能包括：

Model output——绘制模型输出曲线；

Model resids——绘制模型预测残差曲线；

Transient resp——绘制暂态响应曲线；

Frequency resp——绘制频率响应曲线；

Zeros and poles——绘制模型零极点图；

Noise spectrum——绘制噪声频谱。

对于上述的 dryer 模型辨识实例，当采用 2 阶 ARX 模型时，对应的模型预测残差相关函数曲线和频率响应曲线，分别如图 1.7.7 和图 1.7.8 所示。

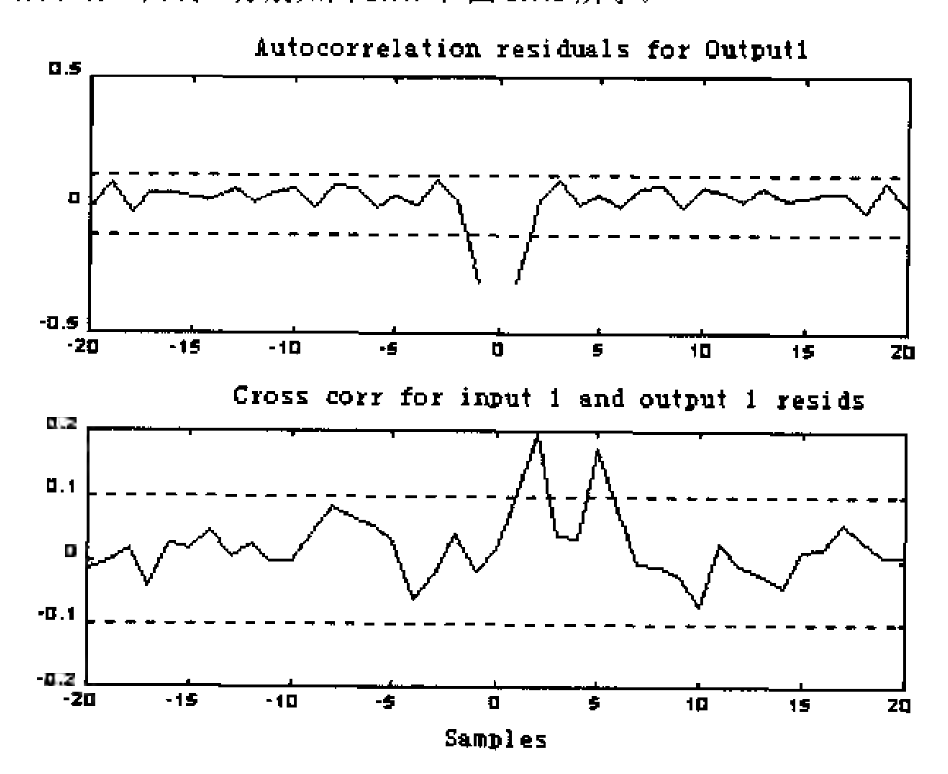


图 1.7.7 ARX 模型预测输出残差的自相关函数与互相关函数曲线

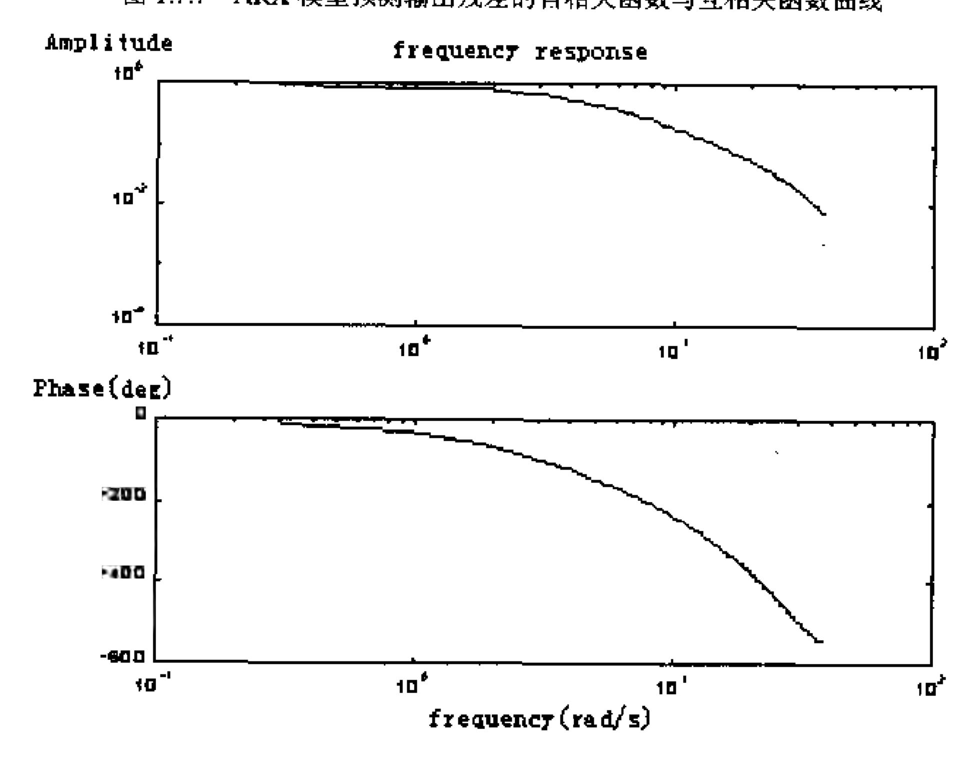


图 1.7.8 ARX 模型频率响应曲线

1.8　系统辨识工具箱的其他功能函数

除了前面介绍的 40 多个函数外，在系统辨识工具箱中还提供了有关模型结构选择和绘图的 10 多个功能函数。这些功能函数可以分为如下几类：

模型结构选择函数；

绘图函数；

模型信息获取函数；

数据的预处理函数；

有关模型不确定性的函数。

下面分别对这些函数进行简单介绍。

1.8.1　模型结构选择函数

系统辨识工具箱中的模型结构选择函数，如表 1.8.1 所示。

表 1.8.1　模型结构选择函数

函数名称	功　能
arxstruc	计算多个 ARX 模型结构的损失函数
ivstruc	计算多个输出误差模型结构的损失函数
selstruc	选择模型结构
struc	生成一系列模型结构

1　模型结构参数的生成

为进行模型结构的选择，往往需要首先生成一系列模型结构以进行比较，函数 struc 用于生成多个 ARX 模型结构参数。

- *struc*

功能：生成多个模型结构参数。

格式：NN = struc(NA,NB,NK)

说明：输入参数定义为：

NA——ARX 模型多项式 A(q)的阶次范围；

NB——ARX 模型多项式 B(q)的阶次范围；

NK——ARX 模型纯时延的大小范围；

上述 NA、NB 和 NK 均为行向量形式。

输出参数定义为：

NN——模型结构参数集构成的矩阵。

举例：

```
NN = struc(1:2,1:2,4:5);
```

2 计算多个 ARX 模型结构的损失函数

函数 arxstruc 用于计算多个 ARX 模型结构的损失函数，即归一化的输出预测误差平方和。

- **arxstruc**

功能：计算多个 ARX 模型结构的损失函数。

格式：v = arxstruc(ze,zv,NN)

v = arxstruc(ze,zv,NN,maxsize)

说明：输入参数定义为：

ze——用于模型辨识的输入输出数据向量或矩阵；

zv——用于模型验证的输入输出数据向量或矩阵；

NN——多个模型结构参数构成的矩阵，NN 的每一行都具有如下的格式：

nn=[na nb nk]

Maxsize——计算辅助变量，参见函数 ar。

输出参数定义为：

v——v 的第一行为各个模型结构的损失函数值，后面的各行为模型结构参数。

在函数 arxstruc 的基础上进行模型结构选择，可以利用函数 selstruc。

举例：

```
NN = struc(1:5,1:5,1);
V = arxstruc(z(1:200,:),z(201:400,:),NN);
```

3 采用辅助变量方法计算 ARX 模型结构的损失函数

函数 ivstruc 用于基于辅助变量方法的 ARX 模型结构损失函数计算。

- **ivstruc**

功能：采用辅助变量方法计算多个 ARX 模型结构的损失函数。

格式：v = ivstruc(ze,zv,NN)

v = ivstruc(ze,zv,NN,p,maxsize)

说明：输入参数定义为：

ze——用于模型辨识的输入输出数据向量或矩阵；

zv——用于模型验证的输入输出数据向量或矩阵；

NN——多个模型结构参数构成的矩阵，NN 的每一行都具有如下的格式：

nn=[na nb nk]

Maxsize——计算辅助变量，参见函数 ar；

p——决定是否计算矩阵的条件数，若 p=0，则不计算矩阵的条件数。

输出参数 v 为对应各个模型结构的损失函数。

举例：

```
A = [1 -0.5 0.7];
B = [0 0 1];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,randn(300,1)],th0);
z = [y,u];
v = ivstruc(z,z,struc(1:3,1:2,0:2))
```

4　在损失函数的基础上进行模型结构选择

函数 selstruc 用于在损失函数的基础上进行模型结构的选择。选择的标准包括 Akaike（赤池）信息理论指标（AIC）和 Rissanen 的最小描述长度指标等。Akaike 信息理论指标（AIC）的定义如下

$$AIC=\log[(1+2n/N)*V]$$

Rissanen 的最小描述长度指标定义为

$$MDL=V*(1+\log(N)*n/N)$$

其中，V 为模型的损失函数即预测误差的平方和，n 为模型参数的个数，N 为数据的长度。

- **selstruc**

功能： 模型结构选择。

格式： [nn,vmod] = selstruc(v)

[nn,vmod] = selstruc(v,c)

说明： 输入参数定义为：

v——各个模型结构的损失函数，其格式定义参见函数 arxstruc 的说明；

c——可选参数，用于指定模型结构选择的方式，具体定义如下：

(1) c='plot'：绘制各个模型结构对应的损失函数值，由用户决定需要的模型结构，为缺省值；

(2) c='log'：绘制各个模型结构对应的损失函数的对数值，由用户决定需要的模型结构；

(3) c='aic'：按照极小化 Akaike 信息理论指标（AIC）选择模型结构；

(4) c='mdl'：按照极小化 Rissanen 的最小描述长度指标选择模型结构；

(5) c=m：m 为一数值量，则模型选择的指标为 $\mathrm{Min}(V*(1+n*m/N))$，其中 V 为损失函数，n 为参数个数，N 为数据长度。

输出参数定义为：

nn——选定的模型结构参数；

vmod——包括模型选择指标对数值的损失函数矩阵，该矩阵的格式定义与 v 相同。

举例： 下面对给定的输入输出数据，在模型结构选择的基础上进行 ARX 模型辨识。

```
A = [1 -0.5 0.7];
B = [0 1 0.5];
```

```
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,randn(300,1)],th0);
z = [y,u];
v1 = arxstruc(z,z,struc(1:3,1:2,0:2));
v2=ivstruc(z,z,struc(1:3,1:2,0:2));
nn1 = selstruc(v1,'aic')
nn2=selstruc(v2,'aic')
```

函数 selstruc 输出的模型结构参数为：

```
nn1=[3 2 1],
nn2=[2 1 0]。
```

1.8.2 系统辨识工具箱的绘图函数

在系统辨识工具箱中提供的绘图函数，如表 1.8.2 所示。

表 1.8.2 绘 图 函 数

函数名称	功 能
bodeplot	绘制波特图
ffplot	绘制频率响应曲线
idplot	显示输入输出数据
nyqplot	绘制 Nyquist 曲线
present	显示模型信息
zpplot	绘制模型的零极点图

1 绘制波特图

- **bodeplot**

功能： 绘制波特图。

格式： bodeplot(g)

bodeplot([g1 g2 ... gn])

bodeplot(g,sd,C,mode)

说明： 该函数有三种调用格式，对应的参数定义为：

g，gi——对象的频率响应数据；

sd——标准偏差，用于绘制频率特性的置信区间；

C——指定绘制幅频特性或相频特性曲线，若 C='A'，则仅绘制幅频特性曲线，若 C='P'，则仅绘制相频特性曲线，若 C='B'，则同时绘制幅频特性和相频特性曲线；

mode——当 mode='same'时，所有曲线均绘制在同一个图中。

举例：首先对一个二阶对象进行基于 4 步辅助变量法的 ARX 模型辨识，然后绘制模型的对数幅频相频特性曲线及其置信区间，如图 1.8.1 所示。

```
v=randn(300,1)
A = [1 -0.5 0.7];
B = [0 1 0.5];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,v],th0);
z = [y,u];
thiv=iv4(z,[2 2 1]);
thiv=th2ff(thiv)
bodeplot(thiv,5,'B','same')
```

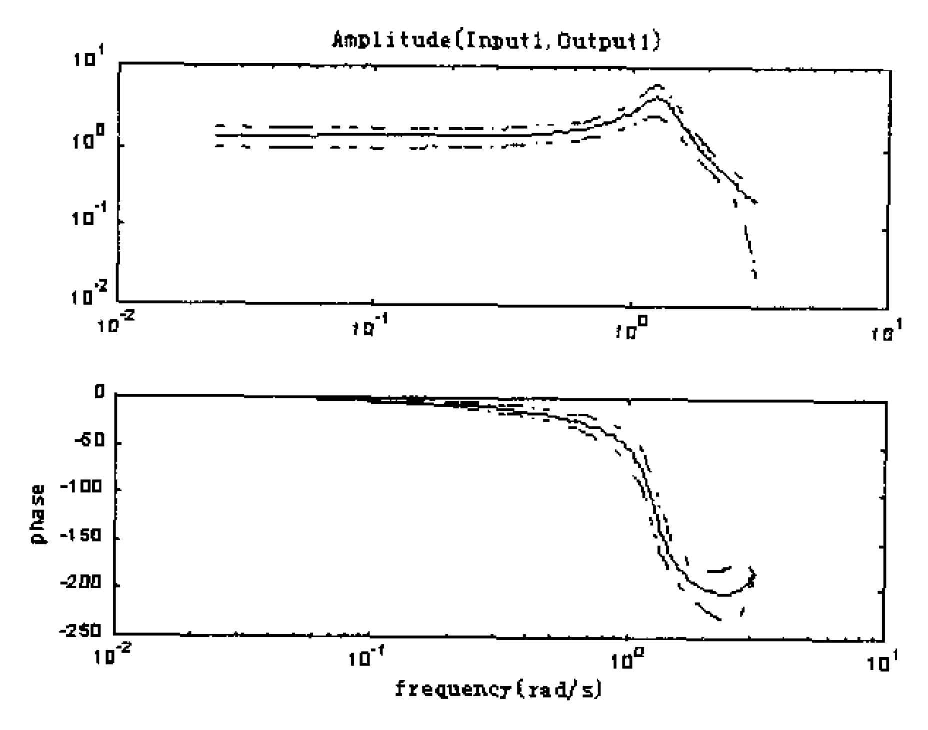


图 1.8.1　模型的频率波特图

2　绘制频率响应曲线

函数 ffplot 用于绘制对象的线性频率特性曲线。

- ffplot

功能：绘制模型的线性频率特性曲线。

格式：ffplot(g)

ffplot([g1 g2 ... gn])

ffplot(g,sd,C,mode)

说明：该函数的输入输出参数定义与函数 bodeplot 相同。

3 绘制对象的输入输出特性曲线

- **idplot**

功能：绘制对象的输入输出特性曲线。

格式：idplot(z)

idplot(z,int,T,ny,pc)

说明：该函数绘制对象的输出曲线和所有输入的变化曲线。在图形窗口中，输出变化曲线在最上方，下面为各个输入的变化曲线。

输入参数定义为：

z——对象的输入输出数据；

int——指定绘制的时间范围；

T——采样周期，缺省值为 1；

ny——输出变量的个数，缺省值为 1；

pc——指定是否在采样间隔之间对输入进行线性插值。若 pc='li'，则进行线性插值；若 pc='pc'，则输入在采样间隔之间保持为常数；缺省值 pc='pc'。

举例：下面的程序在生成对象的输入输出数据后，利用函数 idplot 绘制部分输入输出特性曲线，如图 1.8.2 所示。

```
v=randn(300,1)
A = [1 0.3 0.5]; B = [0 1 -0.5];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,v],th0);
z = [y,u];
idplot(z,100:200);
```

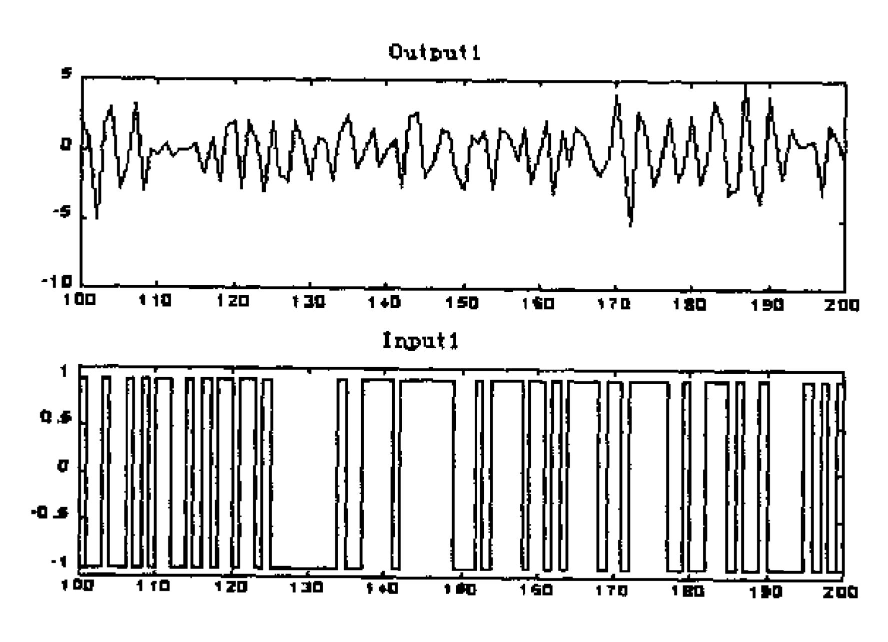


图1.8.2 对象输入输出数据曲线

4 绘制模型的 Nyquist 曲线

函数 nyquist 用于绘制模型的幅相特性曲线即 Nyquist 曲线。

- **nyqplot**

功能：绘制模型的 Nyquist 曲线。

格式：nyqplot(g)

nyqplot([g1 g2 ... gn])

nyqplot(g,sd,mode)

说明：输入输出参数的定义与函数 bodeplot 相同。

举例：绘制对象的 Nyquist 曲线，如图 1.8.3 所示。

```
A = [1 -0.5 0.7];
B = [0 1 0.5];
th0 = poly2th(A,B);
thf=th2ff(th0)
nyqplot(thf)
```

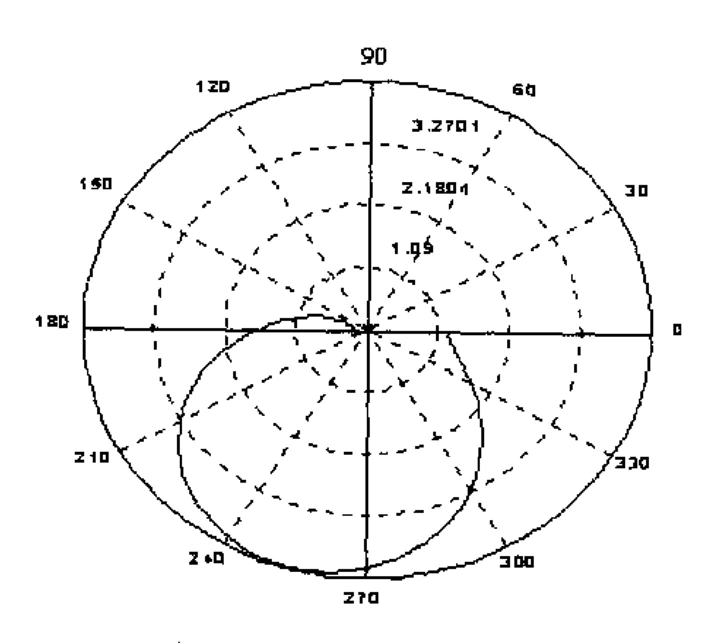


图 1.8.3 对象的 Nyquist 曲线

5 显示模型的有关信息

系统辨识工具箱中的对象模型通常以 Theta 格式矩阵表示，模型的特性参数和数据不易直接获取。利用函数 present 可以较为方便地获得模型的有关信息，包括多项式参数及其标准差、损失函数、AIC 指标值等。

- **present**

功能：显示模型的有关信息。

格式：present(th)

说明：输入参数即为模型的 Theta 格式表示。

举例：

```
v=randn(300,1)
A = [1 -0.5 0.7];
B = [0 1 0.5];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,v],th0);
z = [y,u];
th1=arx(z,[2 2 1]);
th2=iv4(z,[2 2 1]);
present(th1)
present(th2)
```

上述程序运行后显示的模型 th1 的有关信息为：

```
This matrix was created by the command ARX    on 6/22
1999 at 19:47
Loss fcn: 1.0624   Akaike`s FPE: 1.0911  Sampling
interval 1
The polynomial coefficients and their standard
deviations are
B =
        0    1.0432    0.6259
        0    0.0607    0.0676
A =
     1.0000   -0.4702    0.6333
          0    0.0332    0.0302
```

显示模型 th2 的有关信息为：

```
This matrix was created by the command IV4    on 6/22
1999 at 19:47
Loss fcn: 1.0478   Akaike`s FPE: 1.0761  Sampling
interval 1
The polynomial coefficients and their standard
deviations are
B =
        0    1.0537    0.6646
        0    0.0595    0.0749
A =
     1.0000   -0.4225    0.6086
          0    0.0501    0.0417
```

6　绘制模型的零极点图

函数 zpplot 用于绘制模型的零极点图。

- **zpplot**

功能：绘制模型的零极点图。

格式：zpplot(zepo)

zpplot(zpform(zepo1,zepo2, ..., zepon))

zpplot(zepo,sd,mode,axis)

说明：输入参数定义为：

zepo，zepo1,...zepon——由函数 th2zp 得到的对象零极点模型；

zpform(zepo1,...zepon)——将多个零极点模型合并为一个矩阵由函数 zpplot 绘制零极点图；

sd——指定是否绘制置信区间，若 sd 为一个正数，则绘制标准差为 sd 的零极点图置信区间；缺省值为 0；

mode——指定绘制多输入模型的零极点图的绘图模式，mode 的取值有如下三种：

(1)mode='sub'——不同输入的零极点图绘制在不同的子图上；

(2)mode='same'——多个输入的零极点图绘制在同一个图中；

(3)mode='sep'——在绘制下一个输入的零极点图前，删除当前的图形；

axis——指定指标范围，axis=[x1 x2 y1 y2]。

举例：绘制多输入对象的零极点图。

```
A = [1 -0.5 0.7];
B = [0 1 0.5;0 1 0.7];
th1 = poly2th(A,B)
zp1=th2zp(th1)
zpplot(zp1,0,'sub',4)
```

1.8.3　获取模型参数信息的有关函数

在系统辨识工具箱中，各种模型均以矩阵形式表示。为获得有关模型的参数可以利用系统辨识工具箱提供的获取模型信息函数，如表 1.8.3 所示。

表 1.8.3　获取模型信息的功能函数

函数名称	功　能
getff	从频率函数中获得频率响应数据
gett	从 Theta 模型格式中获得采样周期
getmfth	从模型数据矩阵中获得定义模型结构的 M 文件名称
getncap	从 Theta 模型格式中得到该模型对应的测量数据个数
getzp	从零极点模型中获得零极点数据
th2par	从 Theta 模型格式中获得参数估计值和方差

1 从频率模型中获得频率函数

- **getff**

功能：从频率函数中获得频率响应数据。

格式：[w,amp,phas] = getff(g)

[w,amp,phas,sdamp,sdphas] = getff(g,ku,ky)

说明：输入参数定义为：

g——对象的频率函数；

ku,ky——指定多变量系统的某一对输入输出变量，ku=0 则对应噪声输入。

输出参数定义为：

w——频率点向量；

amp——频率响应的幅值；

phas——频率响应的相位；

sdamp,sdphas——幅值和相位的标准差。

举例：

```
v=randn(300,1)
A = [1 -0.5 0.7]; B = [0 1 0.5];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,v],th0);
z = [y,u];
thiv=iv4(z,[2 2 1]);
thiv=th2ff(thiv)
[w,amp,phas]=getff(thiv)
plot(w,amp)
```

2 从 Theta 模型格式中获得有关信息

从对象的 Theta 模型格式中能够获得的有关信息，包括采样周期、数据点个数、定义模型结构的 M 文件名称以及参数估计值和方差等。有关的函数为 gett、getmfth、getncap 和 th2par，这四个函数的使用方法类似，下面简要加以介绍。

- **gett (getmfth,getncap,th2par)**

功能：获得模型采样周期、定义模型结构的 M 文件和测量数据个数等信息。

格式：T = gett(th)

mymfile = getmfth(th)

N = getncap(th)

[par,P,lam] = th2par(th)

说明：输入参数为对象的 Theta 模型格式。

输出参数定义为：

T——模型的采样周期；

mymfile——定义模型结构的 M 文件；

N——测量数据点的个数；

Par——模型的参数估计值；

P——参数估计的方差矩阵；

lam——信息的方差。

3 获得模型的零极点数据

• getzp

功能：获得模型的零极点数据。

格式：[ze,po] = getzp(zepo)

[ze,po] = getzp(zepo,ku,ky)

说明：输入参数定义为：

zepo——对象的零极点模型；

ku,ky——指定多变量系统的某一对输入输出变量；

输出参数定义为：

ze——模型的零点数据；

po——模型的极点数据。

1.8.4 数据的预处理函数

在系统辨识中，数据的预处理如滤波和消除趋势项等具有重要的作用。在系统辨识工具箱中提供的数据预处理函数，如表 1.8.4 所示。

表 1.8.4 数据预处理函数

函数名称	功能
dtrend	消除数据的趋势项
idfilt	对测量数据进行滤波
idresamp	对测量数据进行重新采样

1 消除数据的趋势项

本章的 1.7 节已经介绍了消除趋势项的有关概念，在系统辨识工具箱中完成这一功能的函数为 dtrend。

• dtrend

功能：消除数据的趋势项。

格式：zd = dtrend(z)

zd = dtrend(z,o,brkp)

说明：输入参数定义为：

z——输入输出测量数据；

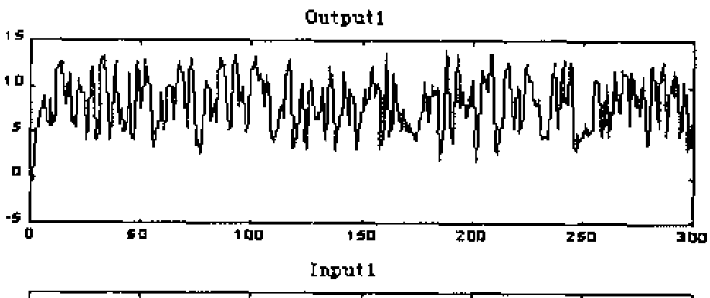
o——趋势项的阶次，缺省值为 o=0，即从数据中减去均值；若 o=1，则从数据中减去线性趋势项（通过线性回归得到）；

brkp——指定分段线性回归的分段点。

输出参数 zd 为消除了趋势项的数据向量或矩阵。

举例：消除输入输出数据的趋势项，原始输入输出如图 1.8.4 所示，消除了趋势项的输入输出数据如图 1.8.5 所示。

```
A=[1 -0.2 0.3] ;
B=[0 1 2]
e=randn(300,1);
th=poly2th(A,B)
u=idinput(300,'rbs',[0 1],[2 4])
y=idsim([u e],th)    ;
z=[y u]
zd=dtrend(z)
```



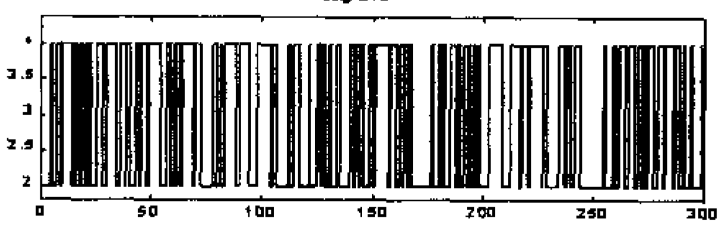


图 1.8.4 原始输入输出数据曲线

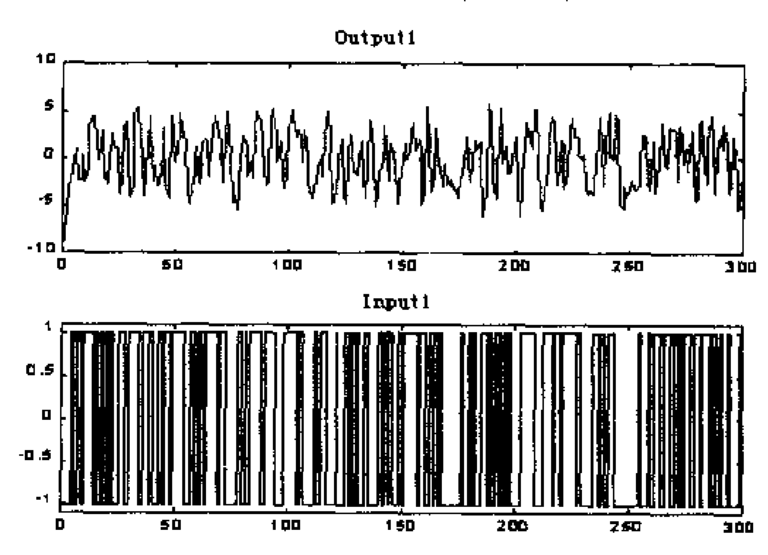


图 1.8.5 消除了趋势项的输入输出数据曲线

2　对测量数据进行滤波

函数 idfilt 对给定的输入输出数据计算相应的 Butterworth 滤波器参数，并对输入输出数据进行滤波。

- **idfilt**

功能：对输入输出数据进行滤波处理。

格式：zf = idfilt(z,ord,Wn)

[zf,thf] = idfilt(z,ord,Wn,hs)

说明：输入参数定义为：

z——输入输出测量数据；

ord——指定 Butterworth 滤波器的阶次；

Wn——当未指定可选参数 hs 时，若 Wn 仅包含一个元素，则该参数用于指定低通滤波器的截止频率；若 Wn=[Wl Wh]，则 Wn 用于指定带通滤波器的上下限频率；当 hs='high'时，Wn 用于指定高通滤波器的截止频率；当 hs='stop'时,Wn=[Wl Wh]用于指定带阻滤波器的频率范围。

输出参数定义为：

zf——滤波后的输入输出数据；

thf——滤波器对应的 Theta 模型格式。

举例：对输入输出数据进行 1 阶高通滤波，滤波后的输入输出数据如图 1.8.6 所示。

```
A=[1 -0.2 0.3]; B=[0 1 2]
e=randn(300,1); th=poly2th(A,B)
u=idinput(300,'rbs',[0 1],[2 4])
y=idsim([u e],th) ; z=[y u];
zf=idfilt(z,2,5,'high')
idplot(zf)
```

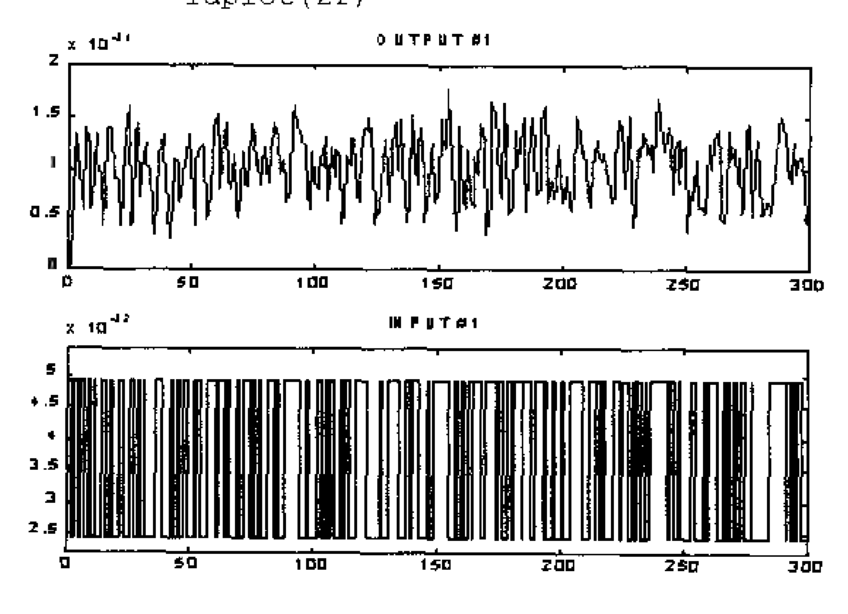


图 1.8.6　高通滤波后的输入输出数据曲线

3 输入输出数据的重新采样

- idresamp

功能：对输入输出数据进行重新采样。

格式：zr = idresamp(z,R)

[zr, R_act] = idresamp(z,R,filter_order,tol)

说明：输入参数定义为：

z——输入输出测量数据；

R——重采样因子，重新采样的周期为原采样周期的 R 倍，R>1，对应数据的均匀删除；R<1，则对应插值过程；

filter_order——指定在进行重新采样之前对数据进行滤波的滤波器阶次，缺省值为 8；

tol——指定对 R 进行有理逼近的精度，缺省值为 0.1。

输出参数定义为：

zr——重新采样后的输入输出数据；

R_act——实际达到的重采样因子。

1.8.5 具有不确定性的模型仿真

由于噪声和模型结构选择等的影响，通过输入输出数据进行系统辨识得到的对象模型也往往存在不确定性。为检验通过系统辨识得到的系统模型在不确定性条件下的响应特性，可以利用函数 idsimsd 进行具有不确定性的模型仿真。

- idsimsd

功能：具有不确定性的模型仿真。

格式：idsimsd(u,th)

idsimsd(u,th,N,noise)

说明：输入参数定义为：

u——仿真输入；

th——对象的 Theta 模型格式；

N——仿真中随机生成的模型个数，这些随机模型的生成按照 th 中的有关模型标准差的数据进行，缺省值为 10；

noise——用于指定是否在仿真中加入噪声；

(1) noise='noise'：加入噪声；

(2) noise='nonoise'：不加入噪声，为缺省值。

举例：

```
v=randn(300,1)
A = [1 -0.5 0.7];
```

```
B = [0 1 0.5];
th0 = poly2th(A,B);
u = idinput(300,'rbs');
y = idsim([u,v],th0);
z = [y,u];
thiv=iv4(z,[2 2 1]);
idsimsd(u,thiv,10,'nonoise')
```

参 考 文 献

[1] Lennart Ljung, *System Identification Toolbox User's Guide*, MathWorks Inc.,1995
[2] 方崇智、萧德云，过程辨识，清华大学出版社，1988

第2章 控制系统工具箱

(Control System Toolbox Ver 4.1)

Matlab 的控制系统工具箱，主要处理以传递函数为主要特征的经典控制和以状态空间为主要特征的现代控制中的问题。该工具箱对控制系统，尤其是 LTI 线性时不变系统的建模、分析和设计提供了一个完整的解决方案。作为 Matlab 最有力和最基本的工具箱之一，它的新版本不仅向后兼容，更重要的是其设计更合理，更易用。概括地说，控制系统工具箱具有以下几个方面的功能。

1 系统建模

新版本的大部分函数同时支持离散时间和连续时间系统，从而更易于使用。

能够建立系统的状态空间、传递函数、零极点增益模型，并可实现任意两者之间的转换；可通过串联、并联、反馈连接及更一般的框图建模来建立复杂系统的模型；可通过多种方式实现连续时间系统的离散化，离散时间系统的连续化及重采样。

藉助于 Matlab 5 的面向对象特性，新版本的控制系统工具箱能够将系统的不同模型统一成不同的对象，从而对每一种模型可以通过单个对象变量来操纵，较旧版本的多个分离变量代表一个模型大大方便了用户。

2 系统分析

不仅支持对 SISO 系统的分析，工具箱也可对 MIMO 系统进行分析。

对系统的频率响应，可支持系统的 Bode 图、Nichols 图，Nyquist 图进行计算和绘制。

对系统的时域响应，可支持对系统的单位阶跃响应、单位脉冲响应、零输入响应以及更广泛的对任意信号进行仿真。提供一个新的 LTI 观测器(LTI Viewer)，大大方便了用户对系统的各种绘制和分析。

3 系统设计

可计算系统的各种特性。如可控和可观 Gramian 矩阵、系统的可控和可观矩阵、传递零点、Lyapunov 方程；频域特性如稳定裕度、阻尼系数以及根轨迹的增益选择等。

支持系统的可控、可观标准型实现、系统的最小实现、均衡实现、降阶实现以及输入延时的 Pade 估计。

可进行系统的极点配置，观测器设计以及 LQ 和 LQG 最优控制等。

2.1　LTI 系统模型及转换

2.1.1　LTI 系统的模型及转换

1　传递函数模型

连续时间动态系统一般以微分方程描述，而 LTI 系统则常以定系数线性常微分方程描述。对于一个连续单输入单输出(single-input/single-output—SISO)系统，设系统的输入为 $u(t)$，输出为 $y(t)$，则系统相应的微分方程为

$$\begin{aligned} & a_1\frac{d^n y(t)}{dt^n}+a_2\frac{d^{n-1}y(t)}{dt^{n-1}}+\cdots+a_n\frac{dy(t)}{dt}+a_{n+1}y(t)= \\ & =b_1\frac{d^m u(t)}{dt^m}+b_2\frac{d^{m-1}u(t)}{dt^{m-1}}+\cdots+b_{m+1}u(t) \end{aligned} \tag{2.1.1}$$

对该方程进行 Laplace 变换，则可得到 SISO 系统的传递函数描述

$$G(s)=\frac{Y(s)}{U(s)}=\frac{b_1s^m+b_2s^{m-1}+\cdots+b_{m+1}}{a_1s^n+a_2s^{n-1}+\cdots+a_ns+a_{n+1}} \tag{2.1.2}$$

离散时间动态系统一般以差分方程描述，而 LTI 系统则常以定系数线性差分方程描述。对于离散 SISO，设采样周期为 T，系统的输入为 $u(i)$，输出为 $y(i)$，则系统相应的微分方程为

$$\begin{aligned} & a_1y(i+n)+a_2y(i+n-1)+\cdots+a_ny(i+1)+a_{n+1}y(i)= \\ & =b_1u(i+m+1)+b_2u(i+m)+\cdots+b_{m+1}u(i) \end{aligned} \tag{2.1.3}$$

对该方程进行 z 变换，则可得到离散 SISO 系统的传递函数描述

$$G(z)=\frac{Y(z)}{U(z)}=\frac{b_1z^m+b_2z^{m-1}+\cdots+b_{m+1}}{a_1z^n+a_2z^{n-1}+\cdots+a_nz+a_{n+1}} \tag{2.1.4}$$

因此，LTI 系统的传递函数可以用两个系数向量来唯一确定：

$$num=[b_1,b_2,\ldots,b_{m+1}]$$
$$den=[a_1,a_2,\ldots,a_n,a_{n+1}]$$

对于多输入多输出的 LTI 系统，系统的传递函数模型为传递函数矩阵。传递函数矩阵有多种表达式，其中最常用的为矩阵分式描述(Matrix Fraction Description 简称 MFD)。

2 状态空间模型

状态方程是描述控制系统的最常用的方式，状态方程可以同时表示 SISO 和 MIMO 系统。

对于连续 LTI 系统，其状态方程描述为

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{2.1.5}$$

对于离散的 LTI 系统，其状态方程描述为

$$\begin{aligned} x(n+1) &= Ax(n) + Bu(n) \\ y(n) &= Cx(n) + Du(n) \end{aligned} \tag{2.1.6}$$

其中 $x(t)$为状态向量，A,B,C,D 为系数矩阵。

因此，系统的状态方程可以用一个矩阵组(A,B,C,D)来表示。

3 零极点模型

零极点模型实际上是传递函数的一种特殊形式，它将系统表示为零点(Zero)、极点(Pole)和增益(Gain)相乘的形式。对于由式(2.1.2)给出的连续系统的传递函数，其零极点模型为

$$G(s) = k\frac{\sum_{i=1}^{m}(s+z_i)}{\sum_{i=1}^{n}(s+p_i)} = k\frac{(s+z_1)(s+z_2)\cdots(s+z_m)}{(s+p_1)(s+p_2)\cdots(s+p_n)} \tag{2.1.7}$$

相应地，对于由式 2.1.4 给出的离散系统的传递函数，其零极点模型为

$$G(z) = k\frac{\sum_{i=1}^{m}(z+z_i)}{\sum_{i=1}^{n}(z+p_i)} = k\frac{(z+z_1)(z+z_2)\cdots(z+z_m)}{(z+p_1)(z+p_2)\cdots(z+p_n)} \tag{2.1.8}$$

其中，k 为系统增益，z_i, i=1,2,...,m 为系统的零点，p_i, i=1,2,...,n 为系统的极点。

因此，系统的零极点模型可用一个增益量、零点向量和极点向量来表示。

4 模型间转换

系统的各种模型之间可以相互转换。例如，对于状态空间模型可以依据下式转换成传递函数模型：

$G(s) = C[sI - A]^{-1}B + D$ (对于连续系统)

$G(z) = C[zI - A]^{-1}B + D$ (对于离散系统)

对于状态空间模型，由于其重要性及存在多种状态空间实现，控制系统工具箱专门提供了一组函数来完成。详见 2.3 节关于状态方程实现。对于其他转换，由于其原理简单，这里不再细述。

2.1.2 LTI 对象

为了避免对一个系统采用多个分离变量进行描述，例如在旧版本的控制系统工具箱中，状态空间模型需要四个分立矩阵来描述。新版本的控制系统工具箱，通过将 LTI 系统的各种模型描述封装成一个 LTI 对象来更好地操纵一个系统。

一个 LTI 对象可以为以下 3 种对象之一：ss 对象，tf 对象和 pzk 对象，它们分别与状态空间模型、传递函数模型、零极点模型相对应。每个对象都具有对象属性和对象方法，同类对象的属性可以继承，通过对象方法可以存取或者设置对象属性值。在控制系统工具箱中，ss 对象、tf 对象和 pzk 对象除了具有一些共同的属性外，还具有一些各自的特有属性。其共同属性见表 2.1.1。

表 2.1.1　模型共有属性列表

属性名	功能	属性值类型
InputName	输入变量名	字符串单元向量
Notes	模型描述	文本
OutputName	输出变量名	字符串单元向量
Ts	采样周期	数值标量
Td	输入延时	数值向量
Userdata	附加数据	任意数据类型

其中，对于采样周期 Ts，当系统为离散系统时，该属性给出了系统的采样周期，Ts=0 表示系统为连续时间系统，Ts=-1 表示离散时间系统的采样周期未定。

输入延时 Td 仅对连续时间系统有效，其值为由每个输入通道的输入延时组成的延时向量。缺省时 Td=0，即无输入延时。

输入变量名 InputName 和输出变量名 OutputName 允许用户定义系统输入输出的名称。其值为一个字符串单元向量，分别与输入和输出具有相同维数，缺省为空。

模型描述 Notes 和附加数据 Userdata 用于存储模型的其他信息。Notes 常用于给出模型描述的文本信息。Userdata 则可以包含用户需要的任意其他数据。缺省时其值为空。

tf 对象的特有属性见表 2.1.2。

表 2.1.2　tf 对象特有属性列表

属性名	功能	属性值类型
den	传递函数分母系数	由数值行向量组成的单元阵列
num	传递函数分子系数	由数值行向量组成的单元阵列
Variable	传递函数变量	's', 'p', 'z', 'q'或者'z^-1'之一

pzk 对象的特有属性见表 2.1.3。

表 2.1.3　pzk 对象特有属性列表

属性名	功能	属性值类型
k	增益	二维矩阵
p	极点	由数值行向量组成的单元阵列
Variable	零极点模型变量	's', 'p', 'z', 'q'或者'z^-1'之一
z	零点	由数值行向量组成的单元阵列

ss 对象的特有属性见表 2.1.4。

表 2.1.4　ss 对象特有属性列表

属 性 名	功　能	属性值类型
a	系数矩阵 A	二维矩阵
b	系数矩阵 B	二维矩阵
c	系数矩阵 C	二维矩阵
d	系数矩阵 D	二维矩阵
e	系数矩阵 E	二维矩阵
StateName	状态变量名	字符串单元向量

其中：tf 对象的 den 和 num 属性，pzk 对象的 k、p、z 属性，ss 对象的 a、b、c、d 属性的意义详见 2.1.1 节。

tf 对象和 pzk 对象的 Variable 属性用于定义传递函数的频率变量。缺省时，连续时间系统为's'(Laplace 变量 s)；离散时间系统为'z' (z 变换变量 z)；替代地可以选择'p'、'q'或者'z^-1'。其主要作用在于传递函数的不同显示。

ss 对象的 e 属性用于定义描述状态空间模型中的系数矩阵，在标准的状态空间模型中 e=I。其中，I 为单位矩阵。

ss 对象的 StateName 属性类似于 InputName 属性，它用于定义状态空间模型中每个状态的名称。

2.1.3　模型建立及模型转换函数

在新版本的控制系统工具箱中，通过 LTI 对象，每种系统模型的生成和模型间转换均可以通过一个函数来实现，新版本的控制系统工具箱中的模型生成和转换函数见表 2.1.5。

表 2.1.5　模型生成及转换函数列表

函数名称	功　能
dss	生成状态空间模型
filt	生成 DSP 形式的离散传递函数
ss	生成状态空间模型或者转换成状态空间模型
tf	生成传递函数模型或者转换成传递函数模型
zpk	生成零极点模型或者转换成零极点模型

1　dss

功能： 生成系统的状态空间(descriptor state-space)模型。

格式： sys = dss(a,b,c,d,e)

sys = dss(a,b,c,d,e,Ts)

sys = dss(a,b,c,d,e,ltisys)

sys = dss(a,b,c,d,e,'Property1',Value1,...,'PropertyN',ValueN)

sys = dss(a,b,c,d,e,Ts,'Property1',Value1,...,'PropertyN',ValueN)

说明： sys = dss(a,b,c,d,e) 生成连续系统的描述状态空间(descriptor state-space)模型为

$$E\dot{x}(t)=Ax(t)+Bu(t)$$
$$y(t)=Cx(t)+Du(t)$$

其中：a,b,c,d,e 分别对于系统的 A,B,C,D,E 参数矩阵，矩阵 e 必须为非奇异。返回值为一个 ss 对象。

sys = dss(a,b,c,d,e,Ts)生成离散系统的描述状态空间(descriptor state-space)模型为

$$Ex(n+1)=Ax(n)+Bu(n)$$
$$y(n)=Cx(n)+Du(n)$$

其中，a,b,c,d,e 分别对应于系统的 A,B,C,D,E 参数矩阵，矩阵 e 必须为非奇异。返回值为一个 ss 对象。采样周期为 Ts。

sys = dss(a,b,c,d,e,ltisys)生成的 ss 对象的其他属性将继承 ltisys 对象的属性。包括采样周期。

sys = dss(a,b,c,d,e,'Property1',Value1,...,'PropertyN',ValueN)和

sys = dss(a,b,c,d,e,Ts,'Property1',Value1,...,'PropertyN',ValueN)

定义 ss 对象的其他属性值，其中 Property1 和 PropertyN 为属性标识字符串。Value1 和 ValueN 为对应的属性值。关于 ss 对象的属性及属性值详见 2.1.2 节。

举例：

```
sys = dss(1,2,3,4,5,'td',0.1,'inputname','voltage',...
   'notes','Just an example')
```

以上命令将生成以下模型

$$5\dot{x}=x+2u$$
$$y=3x+4u$$

2 filt

功能：生成 DSP 形式的离散传递函数。

格式：sys = filt(num,den)

sys = filt(num,den,Ts)

sys = filt(M)

sys = filt(num,den,'Property1',Value1,...,'PropertyN',ValueN)

sys = filt(num,den,Ts,'Property1',Value1,...,'PropertyN',ValueN)

说明：在数字信信号处理中，常常将传递函数表示为 z^{-1} 的形式。例如

$$H(z^{-1})=\frac{1+z^{-1}}{1+2z^{-1}+3z^{-2}}$$

函数 filt 即生成 DSP 格式的传递函数模型。

sys = filt(num,den) 生成 DSP 形式的离散传递函数模型。其中，num 为系统的分子系数，den 为系统的分母系数，返回值为一个 tf 对象。注意，系数向量必

须以 z 的降幂排列。采样周期未定义(sys.Ts= -1)。

sys = filt(num,den,Ts)定义模型的采样周期 Ts，单位为秒。

sys = filt(M) 定义一个增益为 M 的静态系统。

sys = filt(num,den,'Property1',Value1,...,'PropertyN',ValueN)和

sys = filt(num,den,Ts,'Property1',Value1,...,'PropertyN',ValueN) 定义 tf 对象的其他属性值，其中 Property1 和 PropertyN 为属性标识字符串。Value1 和 ValueN 为对应的属性值。关于 tf 对象的属性及属性值详见 2.1.2 节。

使用函数 filt 需要注意以下三点。

- 对于 SISO 系统，num 和 den 为以 z 的降幂排列的系数向量。
- 对于 MIMO 系统，num 和 den 为单元矩阵，其具有与系统输出维数相同的行数及与系统输入维数相同的列数。可将 num 和 den 看作 SISO 系统传递函数矩阵。num(i,j)和 den(i,j)定义了从第 j 个输入到第 i 个输出的传递函数分子分母多项式的系数向量。
- 函数 filt 和 tf 功能相同，只是将 Variable 属性设置为'z^-1'或者'q'。

举例:

```
num = {1 , [1 0.3]}
den = {[1 1 2] ,[5 2]}
H = filt(num,den,'inputname',{'channel1' 'channel2'})
```

以上命令将生成以下模型

$$H(z^{-1})=\begin{bmatrix}\dfrac{1}{1+1z^{-1}+2z^{-2}} & \dfrac{1+0.3z^{-1}}{5+2z^{-1}}\end{bmatrix}$$

3 ss

功能: 生成状态空间模型，或者将传递函数或零极点模型转换成状态空间模型。

格式: sys = ss(a,b,c,d)

sys = ss(a,b,c,d,Ts)

sys = ss(d)

sys = ss(a,b,c,d,ltisys)

sys = ss(a,b,c,d,'Property1',Value1,...,'PropertyN',ValueN)

sys = ss(a,b,c,d,Ts,'Property1',Value1,...,'PropertyN',ValueN)

sys_ss = ss(sys)

说明: sys = ss(a,b,c,d)生成连续系统的状态空间模型。系统描述见式(2.1.5)。其中 a,b,c,d 分别对于系统的 A,B,C,D 参数矩阵。返回值为一个 ss 对象。

sys = ss(a,b,c,d,Ts) 生成离散系统的状态空间模型。系统描述见式(2.1.6)。其中 a,b,c,d 分别对于系统的 A,B,C,D 参数矩阵。Ts 为采样周期，设置 Ts=-1 或者 Ts=[ ] 则系统的采样周期未定义。返回值为一个 ss 对象。

sys = ss(d)等价于 sys = ss([ ],[ ],[ ],d)

sys = ss(a,b,c,d,ltisys) 生成的 ss 对象的其他属性将继承 ltisys 对象的属性，包括

采样周期。

sys = ss(a,b,c,d,'Property1',Value1,...,'PropertyN',ValueN)和

sys = ss(a,b,c,d,Ts,'Property1',Value1,...,'PropertyN',ValueN) 定义 ss 对象的其他属性值，其中 Property1 和 PropertyN 为属性标识字符串。Value1 和 ValueN 为对应的属性值。关于 ss 对象的属性及属性值详见 2.1.2 节。

sys_ss = ss(sys)将任意的 LTI 对象 sys 转换成状态空间模型，即状态实现。

举例：计算下述传递函数的状态空间实现。

$$H(s)=\begin{bmatrix}\dfrac{s+1}{s^3+3s^2+3s+2}\\ \dfrac{s^2+3}{s^2+s+1}\end{bmatrix}$$

```
H = [tf([1 1],[1 3 3 2]) ; tf([1 0 3],[1 1 1])];
ss(H);
a -
x1 x2 x3
x1 -2.00000 0 0.50000
x2 0 -1.00000 -0.50000
x3 0 2.00000 0
b =
u1
x1 0
x2 2.00000
x3 0
c =
x1 x2 x3
y1 -0.50000 0 0.25000
y2 0 -0.50000 0.50000
d =
u1
y1 0
y2 1.00000
Continuous-time system.
```

4　tf

功能：生成传递函数模型，或者将零极点模型或状态空间模型转换成传递函数模型。

格式：sys = tf(num,den)

sys = tf(num,den,Ts)

sys = tf(M)

sys = tf(num,den,ltisys)

sys = tf(num,den,'Property1',Value1,...,'PropertyN',ValueN)

sys = tf(num,den,Ts,'Property1',Value1,...,'PropertyN',ValueN)

tfsys = tf(sys)

tfsys = tf(sys,'inv') % 仅仅用于状态空间模型

说明：sys = tf(num,den) 生成连续时间系统的传递函数模型。系统描述见式(2.1.2)。其中 num,den 分别对于系统的分子，分母系数。返回值为一个 ss 对象。注意，系数向量必须以 z 的降幂排列。

sys = tf(num,den,Ts) 生成连续时间系统的传递函数模型。系统描述见式(2.1.2)。其中 num,den 分别对于系统的分子，分母系数。Ts 为采样周期，设置 Ts=-1 或者 Ts=[ ]则系统的采样周期未定义，返回值为一个 ss 对象。

sys = tf(M) 定义一个增益为 M 的静态系统。

sys = tf(num,den,ltisys) 生成的 tf 对象的其他属性将继承 ltisys 对象的属性，包括采样周期。

sys = tf(num,den,'Property1',Value1,...,'PropertyN',ValueN)和

sys = tf(num,den,Ts,'Property1',Value1,...,'PropertyN',ValueN) 定义 tf 对象的其他属性值，其中 Property1 和 PropertyN 为属性标识字符串。Value1 和 ValueN 为对应的属性值。关于 tf 对象的属性及属性值详见 2.1.2 节。

tfsys = tf(sys) 将任意的 LTI 对象 sys 转换成传递函数模型，缺省时使用 tzero 函数将状态空间模型转换成传递函数模型；使用 poly 函数将零极点增益模型转换成传递函数模型。关于 tzero 函数见 2.4 节。

tfsys = tf(sys,'inv') 使用 pole 函数将状态空间模型转换成传递函数模型。关于 pole 函数见 2.4 节。

使用函数 tf 需要注意以下两点。

- 对于 SISO 系统，num 和 den 为以 z 的降幂排列的系数向量。
- 对于 MIMO 系统，num 和 den 为单元矩阵，其具有与系统输出维数相同的行数及与系统输入维数相同的列数。可将 num 和 den 看作 SISO 系统传递函数矩阵。num(i,j)和 den(i,j)定义了从第 j 个输入到第 i 个输出的传递函数系数向量。

举例：对于下述传递函数，输入命令后 Matlab 将返回以下结果。

$$H(p)=\begin{bmatrix}\dfrac{P+1}{P^2+2P+2}\\ \dfrac{1}{p}\end{bmatrix}$$

```
num = {[1 1] ; 1}
den = {[1 2 2] ; [1 0]}
H = tf(num,den,'inputn','current', outputn',{'torque' 'ang.
velocity'},'variable','p')
Transfer function from input "current" to output...
            p + 1
```

```
torque: -------------------
        p^2 + 2 p + 2

ang. velocity:    1/p
```

5　zpk

功能： 生成零极点模型，或者将状态空间模型或传递函数模型转换成零极点模型。

格式： sys = zpk(z,p,k)

sys = zpk(z,p,k,Ts)

sys = zpk(M)

sys = zpk(z,p,k,ltisys)

sys = zpk(z,p,k,'Property1',Value1,...,'PropertyN',ValueN)

sys = zpk(z,p,k,Ts,'Property1',Value1,...,'PropertyN',ValueN)

zsys = zpk(sys)

zsys = zpk(sys,'inv')　%仅仅用于状态空间模型

说明： sys = zpk(z,p,k) 生成连续时间系统的零极点增益模型。系统描述见式(2.1.7)。其中 z,p,k 分别对于系统的零点、极点和增益。返回值为一个 zpk 对象。

sys = zpk(z,p,k,Ts) 生成离散时间系统的零极点增益模型。系统描述见式(2.1.8)。其中 z,p,k 分别对于系统的零点、极点和增益。Ts 为采样周期，设置 Ts=-1 或者 Ts=[ ]则系统的采样周期未定义。返回值为一个 zpk 对象。

sys = zpk(M) 定义一个增益为 M 的静态系统。

sys = zpk(z,p,k,ltisys) 生成的 zpk 对象的其他属性将继承 ltisys 对象的属性，包括采样周期。

sys = zpk(z,p,k,'Property1',Value1,...,'PropertyN',ValueN)和

sys = zpk(z,p,k,Ts,'Property1',Value1,...,'PropertyN',ValueN) 定义 zpk 对象的其他属性值，其中 Property1 和 PropertyN 为属性标识字符串。Value1 和 ValueN 为对应的属性值。

zsys = zpk(sys) 将任意的 LTI 对象 sys 转换成零极点增益模型。缺省时使用 tzero 函数将状态空间模型转换成传递函数模型；使用 roots 函数将零极点增益模型转换成零极点增益模型。关于 tzero 函数见 2.4 节。

zsys = zpk(sys,'inv') 使用 pole 函数将状态空间模型转换成零极点增益模型。关于 pole 函数见 2.4 节。

举例：

```
z = {[] ; -0.5}
p = {0.3 ; [0.1+i 0.1-i]}
k = [1 ; 2]
H = zpk(z,p,k,-1) % 采样周期未定义
```

以上命令将生成以下模型

$$H(z)=\begin{bmatrix}\dfrac{1}{z-0.3}\\ \dfrac{2(z+0.5)}{(z-0.1+j)(z-0.1-j)}\end{bmatrix}$$

2.1.4 LTI 对象属性的存取和设置

在控制系统工具箱中，用户可以通过以下 3 种方式来设置一个 LTI 对象属性的属性值。

(1) 在使用 tf，zpk，ss，dss，filt 函数生成一个 LTI 对象的同时设置对象属性的属性值。

(2) 使用 set 命令来设置或者修改一个 LTI 对象属性的属性值。

(3) 直接通过对对象属性的赋值来完成。

与设置对象属性相对应，用户可以通过下述 2 种方式来获得一个 LTI 对象属性的属性值。

(1) 使用工具箱提供的函数来获取一个 LTI 对象属性的属性值。

(2) 直接通过对象属性来获得。

通过对象属性直接设置或者存取属性值的方法，实际上是利用对象属性的直接引用来进行的。例如，以下命令可完成对对象属性的操纵。

```
sys = ss(1,2,3,4)
sys.a = -1      %重新设置状态空间模型中的 A=-1
h = tf(1,[1 0],'inputname','u','variable','p');
h.Variable      %获取传递函数模型中 variable 属性的值
ans =
p
```

同时，控制系统工具箱还通过下述函数来支持对对象属性的函数设置和获取，见表 2.1.6。

表 2.1.6 对象属性设置和获取函数列表

函数名称	功能
get	获得 LTI 对象的属性值
set	设置或修改 LTI 对象的属性值
ssdata,dssdata	获得状态空间模型数据
tfdata	获得传递函数模型数据
zpkdata	获得零极点模型数据

1 get

功能：获取 LTI 对象的属性值。

格式：Value = get(sys,'PropertyName')

get(sys)

Struct = get(sys)

说明：Value = get(sys,'PropertyName')获取 LTI 对象 sys 的 Property 属性的属性值 Value。其中，Property 为任意 LTI 对象支持的属性名字符串，可以为属性的

全名(如'UserData')，也可以为大小写不敏感的无歧义的字符串缩写(如'user')。

get(sys) 获取 LTI 对象 sys 的所有属性值。

Struct = get(sys) 将 LTI 对象 sys 转换成标准的 Matlab 结构变量 Struct。其中属性名转换为结构中域的域名，属性值转换为结构中域的值。

举例：生成一个 tf 对象并显示其所有属性值。

```
h = tf(1,[1 2],0.1,'inputname','voltage','user','hello')
get(h)
num = {[0 1]}
den = {[1 2]}
Variable = 'z'
Ts = 0.1
Td = []
InputName = {'voltage'}
OutputName = {''}
Notes = {}
UserData = 'hello'
```

2 set

功能：设置或修改 LTI 对象的属性值。

格式：set(sys,'Property',Value)

set(sys,'Property1',Value1,'Property2',Value2,...)

set(sys,'Property')

set(sys)

说明：set(sys,'Property',Value)设置 LTI 对象 sys 的 Property 属性的属性值为 Value。其中，Property 为任意 LTI 对象支持的属性名字符串，可以为属性的全名(如'UserData')，也可以为大小写不敏感的无歧义的字符串缩写(如'user')。

set(sys,'Property1',Value1,'Property2',Value2,...)同时设置多个属性的属性值。

set(sys,'Property')显示 LTI 对象 sys 的 Property 属性的可能属性值。

set(sys) 显示 LTI 对象 sys 的所有属性及其可能属性值。

举例：生成一个状态空间模型，设置其属性值，查询整个 LTI 对象的属性值。

```
sys = ss(1,2,3,4)
set(sys,'td',0.1, inputn','torque','d',0,'user',...
dcgain(sys))
get(sys)
a = 1
b = 2
c = 3
d = 0
e = []
```

```
StateName = {''}
Ts = 0
Td = 0.1
InputName = {'torque'}
OutputName = {''}
Notes = {}
UserData = -2
```

3 dssdata

功能：获取状态空间模型数据。

格式：[a,b,c,d,e] = dssdata(sys)

[a,b,c,d,e,Ts,Td] = dssdata(sys)

说明：[a,b,c,d,e] = dssdata(sys) 获取 ss 对象 sys 的系数矩阵。其中返回值 a,b,c,d,e 分别与(A,B,C,D,E)对应。如果 sys 是一个 tf 对象或者 zpk 对象，则首先将其转换为 ss 对象。对于一个标准的 ss 对象，返回值 e=I。其中，I 为单位矩阵。

[a,b,c,d,e,Ts,Td] = dssdata(sys)同时返回系统的采样周期 Ts 和输入延时 Td，单位为秒。对于连续时间系统，Td 为与所有输入通道一一对应的延时向量；对于离散时间系统，Td 为空。

4 ssdata

功能：获取传递函数模型数据。

格式：[a,b,c,d] = ssdata(sys)

[a,b,c,d,Ts,Td] = ssdata(sys)

说明：[a,b,c,d] = ssdata(sys) 获取 ss 对象 sys 的系数矩阵。其中返回值 a,b,c,d 分别与(A,B,C,D)对应。如果 sys 是一个 tf 对象或者 zpk 对象，则首先将其转换为 ss 对象。

[a,b,c,d,Ts,Td] = ssdata(sys) 同时返回系统的采样周期 Ts 和输入延时 Td，单位为秒。对于连续时间系统，Td 为与所有输入通道一一对应的延时向量；对于离散时间系统，Td 为空。

5 tfdata

功能：获取传递函数模型数据。

格式：[num,den] = tfdata(sys)

[num,den] = tfdata(sys,'v')

[num,den,Ts,Td] = tfdata(sys)

说明：[num,den] = tfdata(sys)获取 tf 对象 sys 的传递函数分子分母多项式的系数。其中，num 为分子多项式系数，den 为分母多项式系数。num 和 den 均为单元阵列。num(I,j)和 den(I,j)定义了从第 j 个输入到第 i 个输出的传递函数分子分母多项式的系数向量。如果 sys 是一个 ss 对象或者 zpk 对象，则首先将其转换为 tf 对象。

[num,den] = tfdata(sys,'v')仅仅用于 SISO 系统。其返回值不再是单元阵列，而是以行向量返回。

[num,den,Ts,Td] = tfdata(sys) 同时返回系统的采样周期 Ts 和输入延时 Td，单位为秒。对于连续时间系统，Td 为与所有输入通道一一对应的延时向量；对于离散时间系统，Td 为空。

举例：SISO 系统的例子。

```
h = tf([1 1],[1 2 5])
[num,den] = tfdata(h,'v')
num =
0 1 1
den =
1 2 5
```

6　zpkdata

功能：获取零极点模型数据。

格式：[z,p,k] = zpkdata(sys)

[z,p,k] = zpkdata(sys,'v')

[z,p,k,Ts,Td] = zpkdata(sys)

说明：[z,p,k] = zpkdata(sys) 获取 tf 对象 sys 的零点 z、极点 p 和增益 k。其中，z 和 p 均为单元阵列，k 为矩阵。z(i,j)、p(i,j)和 k(i,j)定义了从第 j 个输入到第 i 个输出的零点、极点和增益。如果 sys 是一个 ss 对象或者 tf 对象，则首先将其转换为 zpk 对象。

[z,p,k] = zpkdata(sys,'v') 仅仅用于 SISO 系统。其零点和极点返回值不再是单元阵列，而是以行向量返回。k 为一标量。

[z,p,k,Ts,Td] = zpkdata(sys) 同时返回系统的采样周期 Ts 和输入延时 Td，单位为秒。对于连续时间系统，Td 为与所有输入通道一一对应的延时向量；对于离散时间系统，Td 为空。

工具箱还提供了一组监测模型类型及其他属性，如系统维数等的函数，见表 2.1.7。由于其调用格式简单且类似，这里不再给出详细说明，仅仅列出其调用格式。

表 2.1.7　模型检测函数列表

函数名称	功　能	调用格式
isct	判断 LTI 对象 sys 是否为连续时间系统。若是，返回 1；若不是，返回 0	boo = isct(sys)
isdt	判断 LTI 对象 sys 是否为离散时间系统。若是，返回 1；若不是，返回 0	boo = isdt(sys)
isempty	判断 LTI 对象 sys 是否为空。若是，返回 1；若不是，返回 0	boo = isempty(sys)
isproper	判断 LTI 对象 sys 是否为特定类型对象。若是，返回 1；若不是，返回 0	boo = isproper(sys)
issiso	判断 LTI 对象 sys 是否为 SISO 系统。若是，返回 1；若不是，返回 0	boo = issiso(sys)
size	系统维数计算	

7 size

功能：计算系统的输入、输出或者状态维数。

格式：
```
d = size(sys)
[p,m] = size(sys)
[p,m,n] = size(sys)    % 仅用于 ss 对象
p = size(sys,1)
m = size(sys,2)
n = size(sys,3)    % 仅用于 ss 对象
```

说明：[p,m] = size(sys)返回 LTI 对象 sys 的输入维数 m 和输出维数 p。

d = size(sys)返回 LTI 对象 sys 的输入维数和输出维数。其中，d=[p,m]。

[p,m,n] = size(sys)同时返回 LTI 对象 sys 的状态维数 n，仅用于 ss 对象。

p = size(sys,1)和 m = size(sys,2)和 n = size(sys,3)用于单独查询系统的输入维数 m、输出维数 p 和状态维数 n。查询状态维数仅用于 ss 对象。

2.2 系统建模

对简单系统的建模可以采用上述传递函数、状态空间及零极点模型来直接描述，但实际中常常存在由子系统组成复杂系统的情况，常见的连接模式有并联、串联及反馈连接等。同时考虑到其他一些常用的系统模型，如二阶系统、随机系统等，控制系统工具箱还提供了一组函数来支持这些系统的建模操作。

2.2.1 典型连接

1 系统串联

设系统通过下述两个子系统通过串联实现。

其中，子系统的状态空间模型为：

$$\begin{aligned} \dot{x}_1 &= A_1x_1 + B_1u_1 & \dot{x}_2 &= A_2x_2 + B_2u_2 \\ y_1 &= C_1x_1 + D_1u_1 & y_2 &= C_2x_2 + D_2u_2 \end{aligned} \tag{2.2.1}$$

通过 $u_2=y_1$ 串联，则系统整体模型为：

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} A_1 & 0 \\ B_2C_1 & A_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} B_1 \\ B_2D_1 \end{pmatrix} u_1$$

$$y = \begin{pmatrix} D_2C_1 & C_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + D_2D_1u$$

2 并联

对于如式(2.2.1)的子系统模型，当系统并联时有 $u_1=u_2$，$y=y_1+y_2$，则系统的状态空间模

型为：

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\end{pmatrix}=\begin{pmatrix}A_1 & 0\\ 0 & A_2\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}+\begin{pmatrix}B_1\\ B_2\end{pmatrix}u_1$$

$$y=\begin{pmatrix}C_1 & C_2\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}+(D_2+D_1)u$$

3 反馈连接

反馈连接是系统连接中最常见的形式。对于如式(2.2.1)的子系统模型，设反馈连接具有 $u_2=y_1$，$u_1=r-y_2$，且系统的输入和输出分别为 r 和 y_1，则系统的整体状态空间模型为：

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\end{pmatrix}=\begin{pmatrix}A_1-B_1ZD_2C_1 & -B_1ZC_2\\ B_2(I-D_1ZD_2)C_1 & A_2-B_2D_1ZC_2\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}+\begin{pmatrix}B_1Z\\ B_2D_1Z\end{pmatrix}r$$

$$y=\begin{pmatrix}(I-D_1ZD_2)C_1 & -D_1ZC_2\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}+D_2Zr$$

$$Z=(I-D_1D_2)^{-1}$$

若 $D_1=D_2=0$ 则系统模型简化为：

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\end{pmatrix}=\begin{pmatrix}A_1 & -B_1ZC_2\\ B_2C_1 & A_2\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}+\begin{pmatrix}B_1Z\\ 0\end{pmatrix}r$$

$$y=\begin{pmatrix}C_1 & 0\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}$$

$$Z=(I-D_1D_2)^{-1}$$

4 框图建模

控制系统工具箱还提供了对由多个子系统及多种连接方式建立起来的系统进行建模的函数支持。关于框图建模的方法，请见框图建模函数 connect 的详细说明。

控制系统工具箱对各种典型连接的函数支持见表 2.2.1。

表 2.2.1 系统连接函数列表

函数名称	功能
append	2 个状态空间模型的组合
augstate	系统状态的输出增广
series	系统串联实现
parallel	系统并联实现
feedback	系统反馈连接
connect	框图建模

1 append

功能：多个 LTI 系统的组合。

格式：sys = append(sys1,sys2,...,sysN)

说明：append 函数实现对多个 LTI 系统的组合。对于 N 个子系统 sys1,sys2,…,sysN，组合后的系统见图 2.2.1。

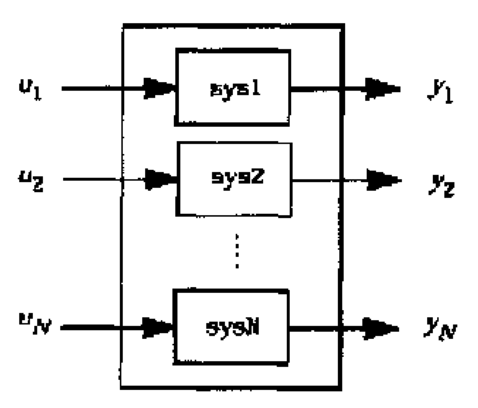


图 2.2.1　系统组合框图

设各子系统的传递函数分别为 $H_1(s)$, $H_2(s)$, …, $H_N(s)$，则合成后的子系统的传递函数为以下分块对角矩阵：

$$\begin{bmatrix} H_1(s) & 0 & \cdots & 0 \\ 0 & H_1(s) & \cdot & \vdots \\ \vdots & \cdot & \cdot & 0 \\ 0 & \cdots & 0 & H_1(s) \end{bmatrix}$$

对于以状态空间模型描述的两个子系统(A_1,B_1,C_1,D_1)和(A_2,B_2,C_2,D_2)，append(sys1,sys2)组合的状态空间描述为：

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} C_1 & 0 \\ 0 & C_2 \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

sys = append(sys1,sys2,...,sysN) 多个 LTI 系统的组合。其中，各个子系统由 sys1,sys2,…,sysN 给出。

举例：
```
sys1 = tf(1,[1 0])
y3 0 0 4.00000
sys2 = ss(1,2,3,4)
sys = append(sys1,10,sys2)
sys
a =
x1 x2
x1 0 0
x2 0 1.00000
b =
```

```
u1 u2 u3
x1 1.00000 0 0
x2 0 0 2.00000
c =
x1 x2
y1 1.00000 0
y2 0 0
y3 0 3.00000
d =
u1 u2 u3
y1 0 0 0
y2 0 10.00000 0
```

2　augstate

功能：将系统的状态增广至系统的输出。

格式：asys = augstate(sys)

说明：给定系统的以下状态空间模型

$$\dot{X} = AX + Bu$$
$$Y = CX + Du$$

则函数 augstate 将系统的状态向量增广至系统的输出中，从而形成系统：

$$\dot{X} = AX + Bu$$
$$\begin{bmatrix} Y \\ X \end{bmatrix} = \begin{bmatrix} C \\ I \end{bmatrix} X + \begin{bmatrix} D \\ 0 \end{bmatrix} u$$

该函数常与函数 feedback 函数一起使用，构成全状态的反馈系统。

asys = augstate(sys) 将系统的状态增广至系统的输出。其中，原系统由 sys 给出。

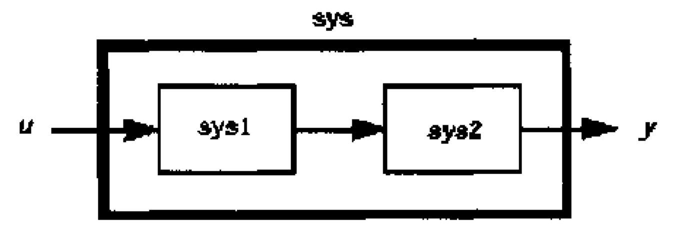


图 2.2.2　系统串联框图

3　series

功能：系统串联实现。

格式：sys = series(sys1,sys2)

sys = series(sys1,sys2,outputs1,inputs2)

说明：sys = series(sys1,sys2) 返回 LTI 对象 sys1 和 sys2 的串联连接系统 sys。两个子系统必须都为连续时间系统或者都为具有相同采样周期的离散时间系统。其串联连接形式如图 2.2.2。

sys = series(sys1,sys2,outputs1,inputs2) 将 sys1 与 sys2 按图 2.2.3 所示的方式连接，在 outputs1 和 inputs1 用于指定 sys1 的部分输出和 sys2 的部分输入进行连接。

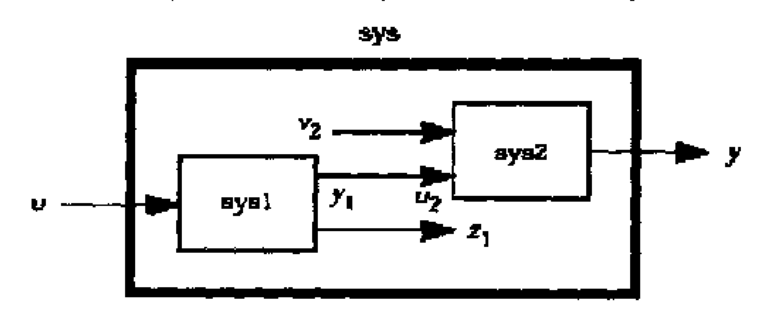


图 2.2.3 系统扩展串联框图

举例：设 sys1 具有 4 输入 4 输出，sys2 具有 3 输入 3 输出，今将 sys1 的输出 2 和输出 4 串联至 sys2 的输入 1 和输入 2，则采用如下命令：

```
outputs1 = [2 4];
inputs2 = [1 2];
sys = series(sys1,sys2,outputs2,inputs1)
```

4 parallel

功能：系统并联实现。

格式：sys = parallel(sys1,sys2)

sys = parallel(sys1,sys2,inp1,inp2,out1,out2)

说明：sys = parallel(sys1,sys2) 返回 LTI 对象 sys1 和 sys2 的并联连接系统 sys。两个子系统必须都为连续时间系统或者都为具有相同采样周期的离散时间系统。其并联连接形式如图 2.2.4。

sys = parallel(sys1,sys2,inp1,inp2,out1,out2) 将 sys1 与 sys2 按图 2.2.5 所示的方式连接，在 inp1 和 inp2 中分别指定两系统连接在一起的输入端编号，out1 和 out2 中分别指定要做相加的输出端编号。这样，系统将具有$[v_1,u,v_2]$输入，$[z_1,y,z_2]$输出。

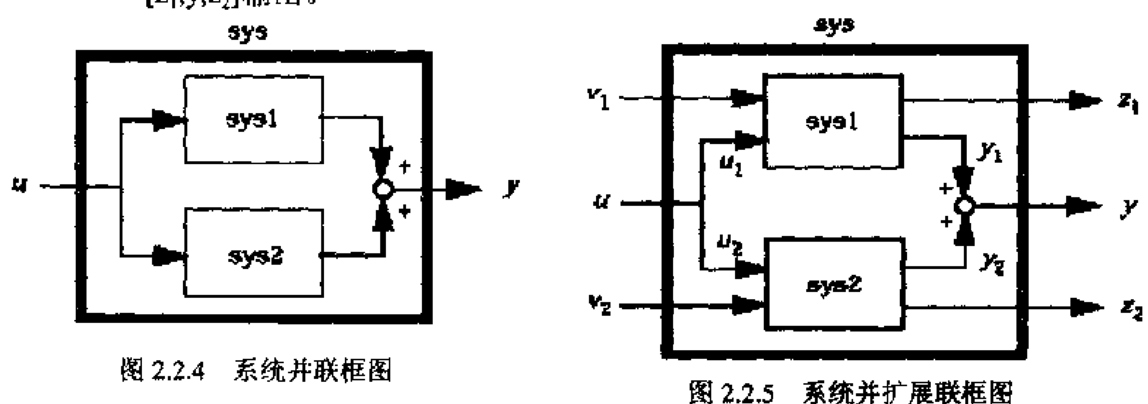


图 2.2.4 系统并联框图

图 2.2.5 系统并扩展联框图

5 feedback

功能： 系统反馈连接。

格式： sys = feedback(sys1,sys2)

sys = feedback(sys1,sys2,sign)

sys = feedback(sys1,sys2,feedin,feedout,sign)

说明： sys = feedback(sys1,sys2)返回 LTI 对象 sys1 和 sys2 的反馈连接系统 sys。两个子系统必须都为连续时间系统或者都为具有相同采样周期的离散时间系统。其反馈连接形式如图 2.2.6。

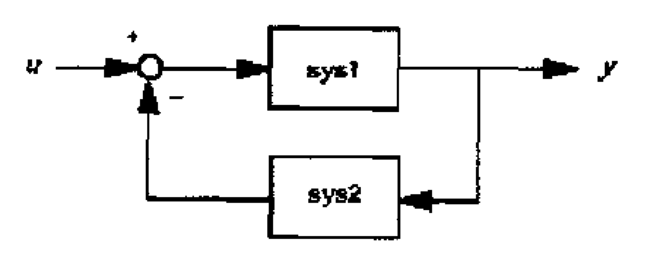


图 2.2.6　系统反馈连接框图

sys = feedback(sys1,sys2,sign)定义反馈形式 sign。如果为正反馈，则 sign=+1；如果为负反馈，则 sign= − 1。

sys = feedback(sys1,sys2,feedin,feedout,sign) 将 sys1 的指定输出 feedout 连接到 sys2 的输入，sys2 的输出连接到 sys1 的指定输入 feedin，以此构成闭环系统，如图 2.2.7 所示。

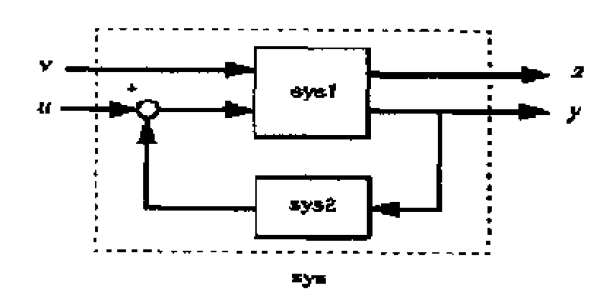


图 2.2.7　系统扩展反馈连接框图

举例： 计算下列反馈系统(见图 2.2.8)。

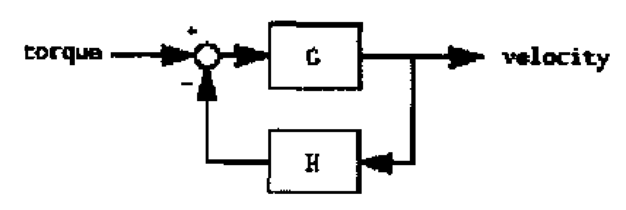


图 2.2.8　反馈系统举例

其传递函数为：

$$G(s)=\frac{2s^2+5s+1}{s^2+2s+3} \qquad H(s)=\frac{5(s+2)}{s+10}$$

```
G=tf([2 5 1],[1 2 3],'inputname','torque',...
'outputname','velocity');
H = zpk(-2,-10,5)
Cloop = feedback(G,H)
```

Matlab 将返回

```
Zero/pole/gain from input "torque" to output  "velocity":
     0.18182 (s+10) (s+2.281) (s+0.2192)
 -----------------------------------------
     (s+3.419) (s^2 + 1.763s + 1.064)
```

6 connect

功能： 框图建模。

格式： sysc = connect(sys,Q,inputs,outputs)

说明： sysc = connect(sys,Q,inputs,outputs)框图建模。其中，sys 通常为由函数 append 生成的无连接对角方块系统，见 append 函数。Q 矩阵用于指定系统 sys 的内部连接关系，inputs 和 outputs 用于选择系统 sysc 的输入和输出。

举例： 以图 2.2.9 的 SISO 传递函数系统模型构成的方框图为例，说明框图建模的过程。

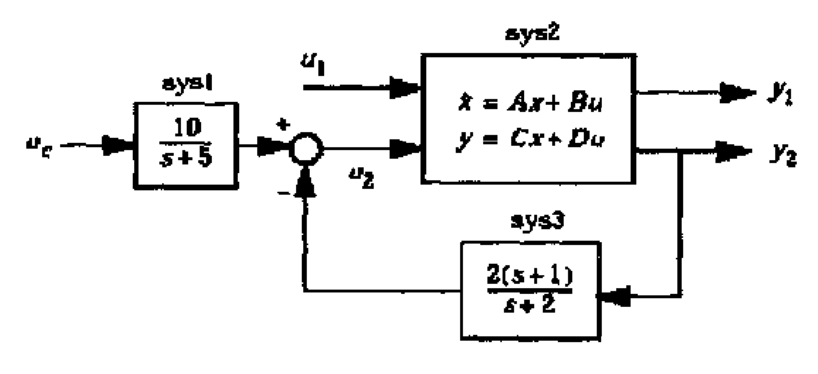


图 2.2.9　框图建模系统

- 定义传递函数或状态空间系统

在方框图上对每个方框进行编号，并输入相应的传递函数分子分母系数：num1, num2, num3, …和 den1, den2, den3,…，或者输入状态矩阵(a, b, c, d)：a1, b1, c1, d1; a2, b2, c2, d2; a3, b3, c3, d3; …，或者定义零极点向量：z1,z2,z3,...和 p1,p2,p3,...及 k1,k2,k3,...；然后利用 ss,tf,zpk 函数对方框图建模。对于给出的例子，有

```
A = [ -9.0201 17.7791
      -1.6943 3.2138 ];
B = [ -55112 .5362
      -.002 -1.8470];
C = [ -3.2897 2.4544
      -13.5009 18.0745];
```

```
D = [-.5476 -.1410
     -.6459 .2958 ];
sys1 = tf(10,[1 5],'inputname','uc )
sys2 = ss(A,B,C,D,'inputname',{'u1  'u2'},...
'outputname',{'y1' 'y2'})
sys3 = zpk(-1,-1,2)
```

• 建立无连接的状态空间模型

形成一个由所有无连接关系传递函数构成的对角块模型(a, b, c, d)，这可以通过重复调用 append 或 tf2ss 和 append 来完成。对于给出的例子，有

```
sys = append(sys1,sys2,sys3)
```

Matlab 将返回

```
sys
a =
     x1  x2       x3      x4
x1   -5  0        0       0
x2   0   -9.0201  17.779  0
x3   0   -1.6943  3.2138  0
x4   0   0        0       -2
b =
     uc  u1       u2       ?
x1   4   0        0        0
x2   0   -0.5112  0.5362   0
x3   0   -0.002   -1.847   0
x4   0   0        0        1.4142
c =
     x1   x2       x3      x4
     2.5  0        0       0
y1   0    -3.2897  2.4544  0
y2   0    -13.501  18.075  0
?    0    0        0       -1.4142
d =
     uc   u1       u2      ?
     0    0        0       0
y1   0    -0.5476  -0.141  0
y2   0    -0.6459  0.2958  0
     0    0        0       2
```

• 指定方框间的连接关系

矩阵 q 用于指示方框间的连接关系。矩阵的每一行对应于一个输入，其第一个元素为输入编号，其后为连接该输入的输出编号，如采用负连接，则以负

值表示。对于给出的例子，有

```
Q = [3 1 -4
     4 3 0];
```

- 选择输入/输出

inputs 和 outputs 用于指示无连接系统中的某些输入/输出保留作为外部的输入/输出。对于给出的例子，有

```
inputs = [1 2];
outputs = [2 3];
```

- 内部连接

调用 sysc = connect(sys,Q,inputs,outputs)这一函数，可以从矩阵 Q 中获得连接信息。对方框图进行连接，从而得到系统 sysc，其输入和输出分别由 inputs 和 outputs 确定。对于给出的例子，有

```
sysc = connect(sys,Q,inputs,outputs)
a =
       x1         x2         x3        x4
x1    -5          0          0         0
x2    0.84223   0.076636   5.6007    0.47644
x3   -2.9012   -33.029    45.164    -1.6411
x4    0.65708  -11.996    16.06     -1.6283
b =
       uc        u1
x1     4         0
x2     0        -0.076001
x3     0        -1.5011
x4     0        -0.57391
c =
       x1          x2         x3        x4
y1   -0.22148   -5.6818    5.6568   -0.12529
y2    0.46463   -8.4826    11.356    0.26283
d =
       uc       u1
y1    0        -0.66204
y2    0        -0.40582
Continuous-time system.
```

2.2.2 典型系统生成

控制系统工具箱中常见系统的生成函数，见表 2.2.2。

表 2.2.2　系统生成函数列表

函数名称	功　能
drmodel, drss	离散时间 n 阶稳定随机系统的生成
ord2	二阶系统生成
pade	系统时间延迟的 pade 近似
rmodel, rss	连续时间 n 阶随机稳定系统的生成

1　drmodel, drss

功能：离散时间 n 阶稳定随机系统的生成。

格式：sys = drss(n)

sys = drss(n,p)

sys = drss(n,p,m)

[num,den] = drmodel(n)

[A,B,C,D] = drmodel(n)

[A,B,C,D] = drmodel(n,p,m)

说明：sys = drss(n)生成一个单输入单输出的 n 阶稳定随机离散时间系统。返回值 sys 为一个 ss 对象。

sys = drss(n,p) 生成一个单输入 p 个输出的 n 阶稳定随机离散时间系统。

sys = drss(n,p,m) 生成一个 m 个输入 p 个输出的 n 阶稳定随机离散时间系统。

[num,den] = drmodel(n) 生成一个单输入单输出的 n 阶稳定随机离散时间系统。以传递函数的分子多项式系数向量 num 和分母多项式系数向量 den 返回。

[A,B,C,D] = drmodel(n) 生成一个单输入单输出的 n 阶稳定随机离散时间系统，以状态空间模型系数矩阵(A,B,C,D)返回。

[A,B,C,D] = drmodel(n,p,m) 生成一个 m 个输入 p 个输出的 n 阶稳定随机离散时间系统，并以状态空间模型系数矩阵(A,B,C,D)返回。

举例：生成一个 2 输入 2 输出的 3 阶稳定随机离散时间系统。

```
sys = drss(3,2,2)
a =
        x1          x2          x3
x1   0.38630    -0.21458    -0.09914
x2  -0.23390    -0.15220    -0.06572
x3  -0.03412     0.11394    -0.22618
b =
        u1          u2
x1   0.98833     0.51551
x2   0           0.33395
x3   0.42350     0.43291
```

```
c =
       x1          x2          x3
y1   0.22595     0.76037     0
y2   0           0           0
d =
       u1          u2
y1   0           0.68085
y2   0.78333     0.46110
Sampling time: unspecified
Discrete-time system.
```

2 ord2

功能：二阶系统生成。

格式：[A,B,C,D] = ord2(wn,z)

[num,den] = ord2(wn,z)

说明：[A,B,C,D] = ord2(wn,z)得到二阶系统的状态空间表示(A,B,C,D)，其中 z 表示阻尼系数 ζ，wn 表示自然频率 ω_n。二阶系统的传递函数可描述如下：

$$H(s)=\frac{1}{s^2+2\zeta\omega_n+\omega_n^2}$$

[num,den] = ord2(wn,z) 以传递函数的分子多项式系数向量 num 和分母多项式系数向量 den 返回。

3 pade

功能：系统时间延迟的 pade 近似。

格式：[num,den] = pade(Td,N)

pade(Td,N)

sysx = pade(sys,N)

sysx = pade(sys,[N1,...,Nm])

说明：pade 函数用于产生时延环节的 n 阶 LTI 逼近模型。

[num,den] = pade(Td,N) 对时延 Td 产生 N 阶 pade 逼近，并以传递函数的分子多项式系数向量 num 和分母多项式系数向量 den 返回。

pade(Td,N)绘制 N 阶 pade 逼近的阶跃响应和频域相位响应以与原理想时延比较。

sysx = pade(sys,N)产生系统 sys 的输入延时的 N 阶 pade 逼近。系统的输入延时可以是不定的，当确定以后，其输入延时将被 N 阶 pade 逼近所替代。

sysx = pade(sys,[N1,...,Nm])定义多输入系统的每个输入通道的输入延时的 N 阶 pade 逼近。

举例：计算一个 0.1 秒延时的 3 阶 pade 逼近，并比较其阶跃响应和频域相位响应，

见图 2.2.10。

```
pade(0.1,3)
```

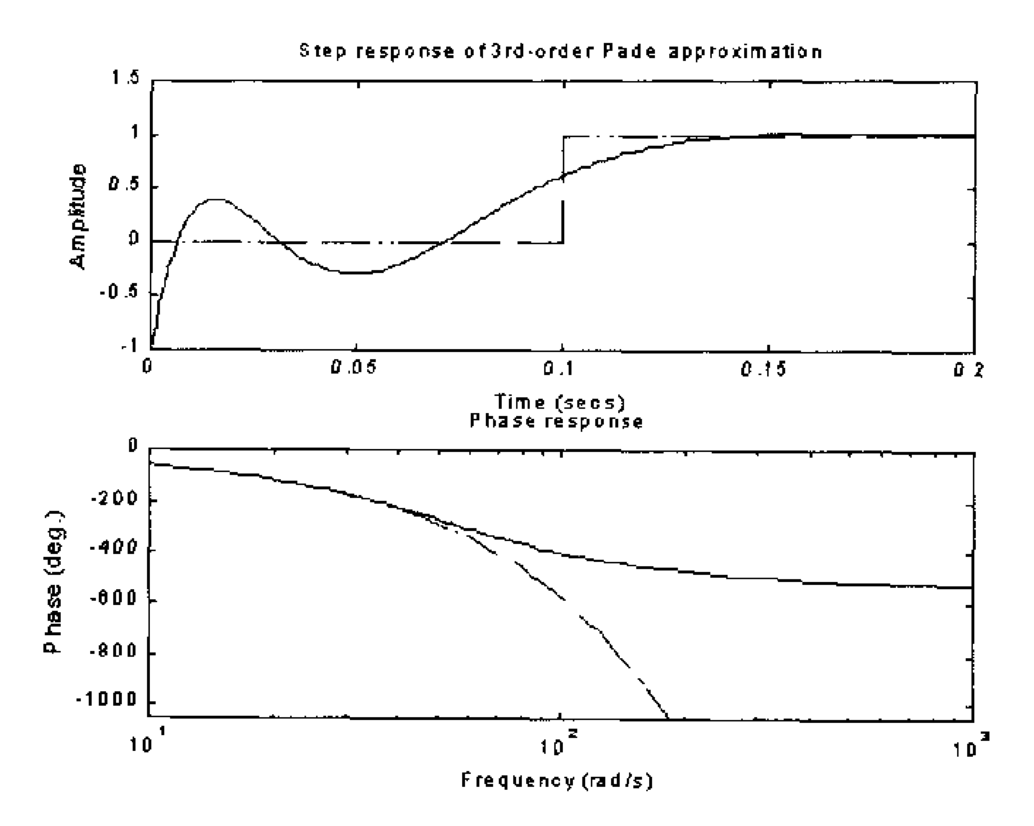


图 2.2.10　系统时延 pade 逼近

4　rmodel

功能： 连续时间 n 阶随机稳定系统的生成。

格式： sys = rss(n)

sys = rss(n,p)

sys = rss(n,p,m)

[num,den] = rmodel(n)

[A,B,C,D] = rmodel(n)

[A,B,C,D] = rmodel(n,p,m)

说明： sys = rss(n) 生成一个单输入单输出的 n 阶稳定随机连续时间系统。返回值 sys 为一个 ss 对象。

sys = rss(n,p) 生成一个单输入 p 个输出的 n 阶稳定随机连续时间系统。

sys = rss(n,p,m) 生成一个 m 个输入 p 个输出的 n 阶稳定随机连续时间系统。

[num,den] = rmodel(n)生成一个单输入单输出的 n 阶稳定随机连续时间系统。以传递函数的分子多项式系数向量 num 和分母多项式系数向量 den 返回。

[A,B,C,D] = rmodel(n) 以状态空间模型系数矩阵(A,B,C,D)返回。

[A,B,C,D] = rmodel(n,p,m) 生成一个 m 个输入 p 个输出的 n 阶稳定随机连续时间系统，并以状态空间模型系数矩阵(A,B,C,D)返回。

举例：生成一个 2 输入 2 输出的 3 阶稳定随机连续时间系统。

```
sys = rss(3,2,2)
a =
      x1          x2          x3
x1  -0.54175    0.09729    0.08304
x2   0.09729   -0.89491    0.58707
x3   0.08304    0.58707   -1.95271
b =
      u1          u2
x1  -0.88844   -2.41459
x2   0         -0.69435
x3  -0.07162   -1.39139
c =
      x1          x2          x3
y1   0.32965    0.14718     0
y2   0.59854   -0.10144     0.02805
d =
      u1          u2
y1  -0.87631   -0.32758
y2   0            0
Continuous-time system.
```

2.2.3　系统的连续化和离散化

连续时间系统的离散化

对于如式(2.1.5)的连续 LTI 系统，设其对应的离散时间系统为

$$\begin{aligned} x(k+1) &= Fx(k)+Gu(k) \\ y(k) &= \hat{C}x(k)+\hat{D}u(k) \end{aligned} \qquad (2.2.2)$$

则有

$$F=e^{At} \quad G=\int_0^T e^{A(T-\tau)}Bd\tau \quad \hat{C}=C \quad \hat{D}=D$$

离散时间系统的连续化

设离散时间 LTI 系统由式(2.2.2)给出，其对应的连续时间 LTI 系统由式(2.1.5)给出，则有

$$A=\tfrac{1}{T}\ln(F) \quad B=(F-I)^{-1}AG \quad C=\hat{C} \quad D=\hat{D}$$

离散时间系统的重采样

在控制系统工具箱中，离散时间系统的重采样通过首先将原系统连续化(采用零阶保持)，然后依据新的采样周期对连续时间系统进行离散化处理来实现(采用零阶保持)。

对系统的连续化和离散化及离散时间系统的重采样的函数见表 2.2.3。

表 2.2.3　系统连续化和离散化函数列表

函数名称	功　能
c2d	连续时间系统离散化
d2c	离散时间系统连续化
d2d	离散时间系统重采样

1　c2d

功能：连续时间系统离散化。

格式：sysd = c2d(sys,Ts)

sysd = c2d(sys,Ts,method)

说明：sysd = c2d(sys,Ts)连续时间 LTI 对象 sys 的离散化。采样周期为 Ts，单位为秒。输入采用零阶保持。

sysd = c2d(sys,Ts,method)定义离散化采用的方法。其中 method 可以为以下字符串之一：

- 'zoh'——采用零阶保持器；
- 'foh'——采用一阶保持器；
- 'tustin'——采用双线性(tustin)逼近方法；
- 'prewarp'——采用改进的 tustin 方法；
- 'matched'——采用 SISO 系统的零极点匹配法。

缺省时，method='zoh'。

举例：系统的传递函数为

$$H(s)=\frac{s-1}{s^2+4s+5}$$

输入延时 Td=0.35 秒，下述命令对系统进行离散化，采用一阶保持，采样周期 Ts=0.1 秒，见图 2.2.11。

```
H = tf([1 -1],[1 4 5],'td',0.35)
Hd = c2d(H,0.1,'foh')
Transfer function:
   0.0115 z^3 + 0.0456 z^2 -00.0562 z -0.009104
---------------------------------------------------
        z^6 -1.629 z^5 + 0.6703 z^4
Sampling time: 0.1
step(H,'-',Hd,'--')
```

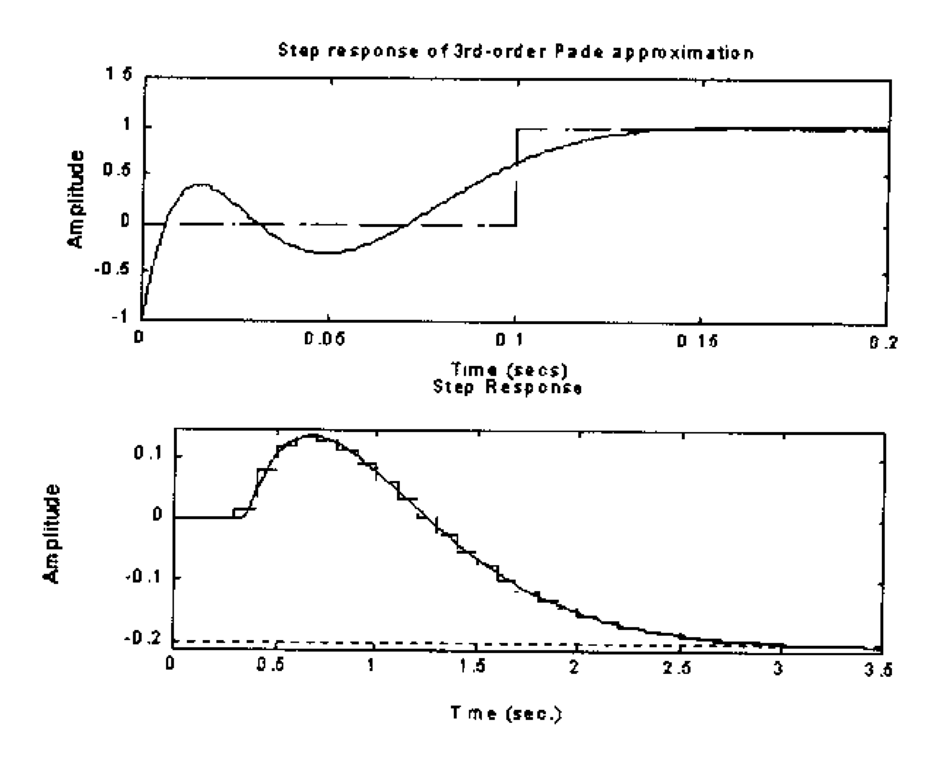


图 2.2.11 连续时间系统离散化

2 d2c

功能：离散时间系统连续化。

格式：sysc = d2c(sysd)

sysc = d2c(sysd,method)

说明：sysc = d2c(sysd) 离散时间 LTI 对象 sysd 的连续化。

sysc = d2c(sysd,method) 定义连续化时采用的方法。其中 method 可以为以下字符串之一：

- 'zoh'——采用零阶保持器；
- 'tustin'——采用双线性(tustin)逼近方法；
- 'prewarp'——采用改进的 tustin 方法；
- 'matched'——采用 SISO 系统的零极点匹配法。

缺省时，method='zoh'。

zoh 方法不适合于系统具有 z=0 的极点的情况。对于具有负实数极点的系统，该方法将增加系统的阶。

tustin 方法不适合于系统具有 z=1 或者具有接近 1 的极点的情况。

举例：一个病态的例子，系统的零极点增益模型为

$$H(z)=\frac{z+0.2}{(z+0.5)(z^2+z+0.4)}$$

采样周期为：Ts=0.1。以下首先连续化，可见系统阶数增加，再进行离散化。

```
Ts = 0.1
```

```
H = zpk(-0.2,-0.5,1,Ts) * tf(1,[1 1 0.4],Ts)
Hc = d2c(H)
```

Matlab 返回

```
Warning: System order was increased to handle real
negative poles.
Zero/pole/gain:
   -33.6556 (s+.273) (s^2 + 28.29s + 1041)
------------------------------------------------
  (s^2 + 9.163s + 637.3) (s^2 + 13.86s + 1035)
```

重新离散化：

```
c2d(Hc,Ts)
```

Matlab 返回

```
Zero/pole/gain:
 (z+0.5) (z+0.2)
-------------------------
 (z+0.5)^2 (z^2 + z + 0.4)
Sampling time: 0.1
```

3 d2d

功能：离散时间系统重采样。

格式：sys1 = d2d(sys,Ts)

sys1 = d2d(sys,[ ],Nd)

说明：sys1 = d2d(sys,Ts) 离散时间 LTI 对象 sys 重采样，从而构成新的离散时间对象 sys1。采样周期为 Ts，单位为秒。该调用等价于命令：sys1 = c2d(d2c(sys),Ts)。

sys1 = d2d(sys,[ ],Nd) 给离散时间 LTI 对象 sys 加入输入延时。输入延时必须是采样周期的整数倍。它由 Nd 给出，如果 Nd 为标量，则各输入通道具有同样的输入延时；如果 Nd 为向量，则分别定义各输入通道的输入延时。

举例：系统传递函数如下，采样周期 Ts=0.1 秒。进行重采样，采样周期 Ts=0.05 秒。

$$H(z)=\frac{z-0.7}{z-0.5}$$

```
H = zpk(0.7,0.5,1,0.1)
H2 = d2d(H,0.05)
```

Matlab 返回

```
Zero/pole/gain:
 (z-0.8243)
-------------
 (z-0.7071)
Sampling time: 0.05
```

2.3　状态空间实现

2.3.1　系统实现

1　相似变换

设 LTI 系统的状态方程由式(2.1.5)给出，现引入非奇异变换矩阵 T，使变换后的状态变量 $x_t(t)=Tx(t)$。设变换后的系统状态方程模型由(A_t, B_t, C_t, D_t)给出。则有

$$A_t = TAT^{-1} \quad B_t = TB \quad C_t CT^{-1} \quad D_t = D$$

该变换成为相似变换，相似变换不影响系统的特征参数和系统结构。

2　系统的能控性、能控标准型实现及系统的能控和不能控分解

设 LTI 系统的状态空间模型可由(A, B, C, D)描述，则系统的能控性可通过 Gram 矩阵判据或者秩判据来判定。

Gram 矩阵判据：LTI 系统完全能控的充要条件是，存在时刻 $t_1>0$，使下述 Gram 矩阵

$$W_c = \int_0^{t_1} e^{A\tau} BB^T e^{A^T\tau} d\tau$$

为非奇异。

秩判据：连续时间 LTI 系统完全能控的充要条件是系统的可控矩阵

$$Co = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}$$

的秩为 n。即 Rank(Co)=n，n 为系统的阶次。

若系统完全能控，则必然存在一个相似变换 T，从而将系统变为能控标准型实现。设变换后的系统状态方程由(A_c, B_c, C_c, D_c) 给出，则矩阵 A_c 具有下述形式：

$$A_c = \begin{pmatrix} 0 & 1 & & \\ \vdots & & \ddots & \\ 0 & & & 1 \\ -a_n & -a_{n-1} & \cdots & -a_1 \end{pmatrix}$$

$$\det(sI - A) = s^n + a_1 s^{n-1} + \ldots + a_{n-1}s + a_n$$

如果系统的阶为 n，而系统的能控矩阵的阶小于 n，则存在一个相似变换 $x_c=Tx$，将系统(A, B, C)进行能控与不能控分解。使系统具有如下格式：

$$\overline{A} = \begin{bmatrix} A_{uc} & 0 \\ A_{21} & A_c \end{bmatrix} \quad \overline{B} = \begin{bmatrix} 0 \\ B_c \end{bmatrix} \quad \overline{C} = \begin{bmatrix} C_{nc} & C_c \end{bmatrix}$$

则(A_c, B_c)构成系统的能控子空间。

3　系统的能观性、能观标准型实现及系统的能观和不能观分解

设 LTI 系统的状态空间模型可由(A, B, C, D)描述，则系统的能观性可通过 Gram 矩阵判据或者秩判据来判定。

Gram 矩阵判据：LTI 系统完全能观的充要条件是，存在时刻 $t_1>0$，使下述 Gram 矩阵

$$W_o = \int_0^{t_1} e^{A^T\tau} C^T C e^{A\tau} d\tau$$

为非奇异。

秩判据：连续时间 LTI 系统完全能观的充要条件是系统的可观矩阵

$$Ob = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

的秩为 n。即 $\mathrm{Rank}(Ob)=n$，n 为系统的阶次。

若系统完全能观，则必然存在一个相似变换 T，从而将系统变为能观标准型实现。设变换后的系统状态方程由(A_o, B_o, C_o, D_o) 给出，则矩阵 A_o 具有下述形式：

$$A_o = \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 & -a_n \\ 1 & 0 & 0 & \cdots & 0 & -a_{n-1} \\ 0 & 1 & 0 & . & \vdots & \vdots \\ \vdots & 0 & . & . & \vdots & \vdots \\ 0 & . & . & 1 & 0 & -a_2 \\ 0 & \cdots & \vdots & 0 & 1 & -a_1 \end{bmatrix}$$

$$\det(sI - A) = s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n$$

如果系统的阶为 n，而系统的能观测矩阵的阶小于 n，则存在一个相似变换 $x_o=Tx$，将系统(A, B, C)进行能观与不能观分解。使系统具有如下格式：

$$\bar{A} = \begin{bmatrix} A_{no} & A_{12} \\ 0 & A_o \end{bmatrix} \qquad \bar{B} = \begin{bmatrix} B_{no} \\ B_o \end{bmatrix} \qquad \bar{C} = \begin{bmatrix} 0 & C_o \end{bmatrix}$$

则(A_o, C_o)构成系统的能观子空间。

4　系统的Jordan标准型实现

Jordan 标准型是通过一个相似变换 T，使原系统的 A 矩阵变为

$$\Lambda = T^{-1}AT = \begin{pmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_k \end{pmatrix}$$

Λ 称为矩阵 A 的模态矩阵(model matrix)，而 J_i 称为系统的 Jordan 子矩阵。且有

$$J_i = \begin{pmatrix} \lambda_i & 1 & 0 & \cdots & 0 \\ 0 & \lambda_i & 1 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & 0 & \cdots & \lambda_i \end{pmatrix}$$

λ_i 为矩阵 A 的特征值。

5 系统的可控可观分解及最小实现

通过相似变换，可将系统(A, B, C, D)等效地变换为下述规范形式：

$$\dot{\bar{x}} = \begin{pmatrix} \hat{A}_{C,O} & 0 & \hat{A}_{1,3} & 0 \\ \hat{A}_{2,1} & \hat{A}_{C,\bar{O}} & \hat{A}_{2,3} & \hat{A}_{2,4} \\ 0 & 0 & \hat{A}_{\bar{C},O} & \\ 0 & 0 & \hat{A}_{4,3} & \hat{A}_{\bar{C},\bar{O}} \end{pmatrix} \bar{x} + \begin{pmatrix} \hat{B}_{C,O} \\ \hat{B}_{C,\bar{O}} \\ 0 \\ 0 \end{pmatrix} u$$

$$y = \begin{pmatrix} \hat{C}_{C,O} & 0 & \hat{C}_{\bar{C},O} & 0 \end{pmatrix} + \hat{D}u$$

式中：

$(\hat{A}_{C,O}, \hat{B}_{C,O}, \hat{C}_{C,O})$ 为系统的可控可观子空间

$(\hat{A}_{C,\bar{O}}, \hat{B}_{C,\bar{O}}, 0)$ 为系统的可控不可观子空间

$(\hat{A}_{\bar{C},O}, 0, \hat{C}_{\bar{C},O})$ 为系统的不可控可观子空间

$(\hat{A}_{\bar{C},\bar{O}}, 0, 0)$ 为系统的不可控不可观子空间

对系统的上述分解称为系统的可控可观分解，又称 Kalman 分解。在可控可观分解中，可控可观子空间称为原始系统的最小实现。

6 系统的均衡实现

系统的均衡实现可以改变系统内各状态变量的内部坐标标度，从而使系统不会因变量间标度相差悬殊而带来计算误差。

连续时间系统(A, B, C, D)的可控和可观 Gramian 矩阵为：

$$W_c = \int_0^\infty e^{A\tau} BB^T e^{A^T\tau} d\tau \qquad W_o = \int_0^\infty e^{A^T\tau} C^T C e^{A\tau} d\tau$$

可以证明，W_c，W_o 均为对称的半正定矩阵，且满足 Lyapunov 方程：

$$AW_c + W_c A^T + BB^T = 0$$

$$A^T W_o + W_o A + C^T C = 0$$

从而存在一个相似变换，可将系统转换为下述形式：

$$\dot{x}_b = A_b x_b + B_b u$$

$$y = C_b x + D_b u$$

且系统满足 Lyapunov 方程：

$$A_b \Sigma + \Sigma A_b{}^T + B_b B_b{}^T = 0$$

$$A_b{}^T \Sigma + \Sigma A_b + C_b{}^T C_b = 0$$

其中，Σ 为对角矩阵，可见变换后系统的可控与可观 Gram 矩阵为相等的对角矩阵 Σ。

系统(A_b, B_b, C_b, D_b)为系统的内部均衡实现。

7　系统的降阶实现

模型降阶的目的是使高阶系统可以由一个相对低阶的模型来近似。

常用的模型降阶技术有主导特征值法、奇异摄动法、均衡实现法等，在控制系统工具箱中提供了基于均衡实现的降阶实现函数。

设系统的均衡实现中可控与可观 Gram 矩阵为 Σ，则它可被分解为 $\Sigma=\mathrm{diag}(\Sigma_1, \Sigma_2)$。其中 Σ_1 包含了原系统的较大特征值，Σ_2 包含了原系统的较小特征值，从而相应地将系统分解为：

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} = \begin{pmatrix} A_{b,11} & A_{b,12} \\ A_{b,21} & A_{b,22} \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} B_{b,1} \\ B_{b,2} \end{pmatrix} u$$

$$y = \begin{pmatrix} C_{b,1} & C_{b,2} \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$$

截掉对应于小奇异值的子系统 Σ_2，则降阶模型可以写成：

$$\dot{z}_1 = A_{b,11} z_1 + B_{b,1} u$$

$$\hat{y} = C_{b,1} z_1 + Du$$

但这种降阶模型无法保证系统的稳定性，可将降阶系统改写为：

$$\dot{x}_1 = \left(A_{b,11} - A_{b,12} A_{b,22}^{-1} A_{b,21}\right) x_1 + \left(B_{b,1} - A_{b,12} A_{b,22}^{-1} B_{b,2}\right)$$

$$y = \left(C_{b,1} - A_{b,12} A_{b,22}^{-1} A_{b,21}\right) x_1 + \left(D - C_{b,2} A_{b,22}^{-1} B_{b,2}\right) u$$

来保证系统稳定。但由于前馈矩阵的存在，降阶系统的初始响应值可能与原系统不同。

2.3.2　状态空间实现函数

控制系统工具箱中各种状态空间实现的函数，见表 2.3.1。

表 2.3.1　状态空间实现函数列表

函数名称	功　能
ss2ss	相似变换
canon	状态空间的正则实现
ctrb	可控矩阵计算
ctrbf	系统的可控与不可控分解
gram	求系统的可控与可观 gramian 矩阵
obsv	可观矩阵计算
obsvf	系统的可观与不可观分解
ssbal	状态空间的对角均衡实现
balreal	状态空间的均衡实现
minreal	状态空间的最小实现
modred	模型降阶

1　ss2ss

功能： 相似变换。

格式： sysT = ss2ss(sys,T)

说明： ss2ss 可完成系统的相似变换。相似变换即对于如式(2.1.5)系统，进行 $x_T=Tx$ 变换，变换后的系统为

$$\dot{X}_T = TAT^{-1}X_T + TBu$$
$$Y = CT^{-1}X_T + Du$$

sysT = ss2ss(sys,T) 完成上述状态空间 LTI 对象 sys 的相似变换。其中，T 为变换矩阵。

2　canon

功能： 状态空间的正则实现。

格式： csys = canon(sys,'type')

[csys,T] = canon(sys,'type')

说明： canon 函数计算连续时间或者离散时间 LTI 对象的正则实现。该函数支持两种正则形式：模态矩阵(modal matrix)和伴随矩阵(companion matrix)形式。

- 模态矩阵形式

设系统的特征值为(λ_1, $\sigma \pm j\omega$, λ_2)，则模态矩阵 A 为

$$\begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \sigma & \omega & 0 \\ 0 & -\omega & \sigma & 0 \\ 0 & 0 & 0 & \lambda_2 \end{bmatrix}$$

• 伴随矩阵形式

设系统的特征多项式为

$$p(s)=s^n+a_1s^{n-1}+\ldots+a_{n-1}s+a_n$$

则伴随矩阵为

$$\begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 & -a_n \\ 1 & 0 & 0 & \cdots & 0 & -a_{n-1} \\ 0 & 1 & 0 & . & \vdots & \vdots \\ \vdots & 0 & . & . & \vdots & \vdots \\ 0 & . & . & 1 & 0 & -a_2 \\ 0 & \cdots & \vdots & 0 & 1 & -a_1 \end{bmatrix}$$

csys = canon(sys,'type')计算 LTI 对象 sys 的正则实现。如果系统不是以状态空间模型给出，则首先进行模型转换。其中可以为以下字符串之一：

'modal'——计算模态矩阵正则实现；

'companion'——计算伴随矩阵正则实现。

[csys,T] = canon(sys,'type')同时返回相似变换矩阵 T。当给出的 LTI 对象不是 ss 对象时，T 为空矩阵。

注意：1. 模态实现要求系统的 A 矩阵为可对角化的。

2. 伴随实现需要系统关于第一个输入为可控的，应尽量避免求伴随实现。

3　ctrb

功能：可控矩阵计算。

格式：Co = ctrb(A,B)

Co = ctrb(sys)

说明：ctrb 函数用于计算 LTI 系统的可控矩阵(controllability matrix)。对于一个 n*n 的矩阵 A 及一个 n*m 的矩阵 B。其可控矩阵为

$$Co=\begin{bmatrix}B & AB & A^2B & \cdots & A^{n-1}B\end{bmatrix}$$

如果 Rank(Co)=n，则系统可控。

Co = ctrb(A,B)计算由矩阵 A 和矩阵 B 给出的系统的可控矩阵 Co。

Co = ctrb(sys)计算状态空间 LTI 对象的可控矩阵 Co。该调用等价于 Co = ctrb(sys.A,sys.B)。

4　ctrbf

功能：系统的可控与不可控分解。

格式：[Abar,Bbar,Cbar,T,k] = ctrbf(A,B,C)

[Abar,Bbar,Cbar,T,k] = ctrbf(A,B,C,tol)

说明：如果系统的阶为 n，而系统的可控矩阵的阶小于 n，则存在一个相似变换 $x_c=Tx$，将系统(A,B,C)进行可控与不可控分解。使系统具有如下格式：

$$\overline{A}=\begin{bmatrix} A_{uc} & 0 \\ A_{21} & A_c \end{bmatrix} \qquad \overline{B}=\begin{bmatrix} 0 \\ B_c \end{bmatrix} \qquad \overline{C}=\begin{bmatrix} C_{nc} & C_c \end{bmatrix}$$

则(A_c,B_c)构成系统的可控子空间。

[Abar,Bbar,Cbar,T,k] = ctrbf(A,B,C) 将系统分解为可控/不可控两部分。如上所述，T 为相似变换，k 是长度为 n 的一个矢量，其元素为各个块的秩。sum(k)可求出 A 中可控部分的秩。(Abar,Bbar,Cbar)对应于转换后系统的(A,B,C)。

[Abar,Bbar,Cbar,T,k] = ctrbf(A,B,C,tol) 定义误差容限 tol，缺省时 tol=10*n*norm(a, 1)*eps。

举例：

```
A =
1 1
4 -2
B =
1 -1
1 -1
C =
1 0
0 1
[Abar,Bbar,Cbar,T,k]=ctrbf(A,B,C)
Abar =
-3.0000 0
-3.0000 2.0000
Bbar =
0.0000 0.0000
1.4142 -1.4142
Cbar =
-0.7071 0.7071
0.7071 0.7071
T =
0.7071 0.7071
0.7071 0.7071
k =
1 0
```

系统分解表明，该系统有一个可控状态和一个不可控状态。

5 gram

功能：求系统的可控与可观 gramian 矩阵。

格式：Wc = gram(sys,'c')

　　　Wo = gram(sys,'o')

说明：gram 函数计算系统的可控与可观 gramian 矩阵。gramian 矩阵可用于研究系统的可控与可观性。该函数较 ctrb 和 obsv 函数有更好的属性。

对于由式(2.1.5)给出的连续时间系统，其可控 gramian 矩阵 W_c 与可观 gramian 矩阵 W_o 为

$$W_c = \int_0^{\infty} e^{A\tau} BB^T e^{A^T\tau} d\tau \qquad W_o = \int_0^{\infty} e^{A^T\tau} C^T C e^{A\tau} d\tau$$

对于由式(2.1.6)给出的离散时间系统，其可控 gramian 矩阵 W_c 与可观 gramian 矩阵 W_o 为

$$W_c = \sum_{k=0}^{\infty} A^k BB^T \left(A^T\right)^k \qquad W_o = \sum_{k=0}^{\infty} \left(A^T\right)^k C^T C A^k$$

且可控 gramian 矩阵负定与系统可控等价，可观 gramian 矩阵负定与系统可观等价。

Wc = gram(sys,'c')计算 LTI 对象 sys 的可控 gramian 矩阵

Wo = gram(sys,'o')计算 LTI 对象 sys 的可观 gramian 矩阵

算法：W_c 可通过求解下述连续时间 Lyapunov 方程

$$AW_c + W_c A^T + BB^T = 0$$

或者离散时间 Lyapunov 方程

$$AW_c A^T - W_c + BB^T = 0$$

W_o 可通过求解下述连续时间 Lyapunov 方程

$$A^T W_o + W_o A + C^T C = 0$$

或者离散时间 Lyapunov 方程

$$A^T W_o A - W_o + C^T C = 0$$

6 obsv

功能：可观矩阵计算。

格式：Ob = obsv(A,B)

　　　Ob = obsv(sys)

说明：obsv 函数用于计算 LTI 系统的可观矩阵(observability matrix)。对于一个 n*n 的矩阵 A 及一个 p*n 的矩阵 B。其可观矩阵为

$$Ob=\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

如果 Rank(Ob)=n，则系统可控。

Ob = obsv(A,B) 计算由矩阵 A 和矩阵 B 给出的系统的可观矩阵 Ob。

Ob = obsv(sys) 计算状态空间 LTI 对象的可观矩阵 Ob。该调用等价于 Ob = obsv(sys.A,sys.C)。

7 obsvf

功能：系统的可观与不可观分解。

格式：[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C)

[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C,tol)

说明：如果系统的阶为 n，而系统的可观矩阵的阶小于 n，则存在一个相似变换 $x_o=Tx$，将系统(A,B,C)进行可观与不可观分解。使系统具有如下格式：

$$\overline{A}=\begin{bmatrix} A_{no} & A_{12} \\ 0 & A_o \end{bmatrix} \qquad \overline{B}=\begin{bmatrix} B_{no} \\ B_o \end{bmatrix} \qquad \overline{C}=\begin{bmatrix} 0 & C_o \end{bmatrix}$$

则(A_o,C_o)构成系统的可观子空间。

[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C) 将系统分解为可观/不可观两部分。如上所述，T 为相似变换，k 是长度为 n 的一个矢量，其元素为各个块的秩。sum(k) 可求出 A 中可观部分的秩。(Abar,Bbar,Cbar)对应于转换后系统的(A,B,C)。

[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C,tol) 定义误差容限 tol，缺省时，

tol=10*n*norm(a, 1)*eps

举例：
```
A =
1 1
4 -2
B =
1 -1
1 -1
C =
1 0
0 1
[Abar,Bbar,Cbar,T,k] = obsvf(A,B,C)
Abar =
1 1
4 -2
```

```
Bbar =
1 1
1 -1
Cbar =
1 0
0 1
T =
1 0
0 1
k =
2 0
```

8　ssbal

功能： 状态空间的均衡实现。

格式： [sysb,T] = ssbal(sys)

[sysb,T] = ssbal(sys,condT)

说明： ssbal 将系统(A,B,C)进行相似变换 $x_T=Tx$，变换矩阵 T 具有下述形式

$$\begin{bmatrix} TAT^{-1} & TB/\alpha \\ \alpha CT^{-1} & 0 \end{bmatrix}$$

该矩阵的行和列具有近似相等的范数。从而得到系统的均衡实现

$$\left(TAT^{-1} \quad TB/\alpha \quad \alpha CT^{-1} \quad D\right)$$

[sysb,T] = ssbal(sys) 计算状态空间的均衡实现。

[sysb,T] = ssbal(sys,condT)同时定义相似转换矩阵 T 的范数上界 condT。缺省值为 condT=1/eps。

举例： 实现下述系统的均衡实现。

$$A=\begin{bmatrix} 1 & 10^4 & 10^2 \\ 0 & 10^2 & 10^5 \\ 10 & 1 & 0 \end{bmatrix} \quad B=\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad C=\begin{bmatrix} 0.1 & 10 & 100 \end{bmatrix}$$

```
a = [1 1e4 1e2;0 1e2 1e5;10 1 0];
b = [1;1;1];
c = [0.1 10 1e2];
sys = ss(a,b,c,0)
ssbal(sys)
a =
        x1          x2            x3
x1  1.00000   2500.00000   0.39062
```

```
x2  0        100.00000   1562.50000
x3  2560.00  64.00000    0
b =
      u1
x1 0.12500
x2 0.50000
x3 32.00000
c =
    x1        x2         x3
    y1 0.80000  20.00000  3.12500
    d =
        u1
    y1  0
    Continuous-time system
```

9 balreal

功能: 状态空间的均衡实现。

格式: sysb = balreal(sys)

[sysb,g,T,Ti] = balreal(sys)

说明: sysb = balreal(sys)实现 LTI 对象 sys 的均衡实现。这种均衡模型常常用于模型的降价。该函数可同时用于连续或者离散系统。如果系统不是以状态空间模型给出，则首先进行模型转换。函数返回为均衡实现的 LTI 对象 sysb。

[sysb,g,T,Ti] = balreal(sys)同时返回平衡化参数 g，g 为 LTI 对象均衡实现的相等的对角可控和可观 gramians 矩阵。矩阵 T 为相似变换矩阵，T 完成 $x_b=Tx$。Ti=T^{-1}。

举例:

```
sys = zpk([-10 -20.01],[-5 -9.9 -20.1],1)
Zero/pole/gain:
    (s+10) (s+20.01)
 --------------------------
  (s+5) (s+9.9) (s+20.1)
```

系统的均衡实现

```
[sysb,g] = balreal(sys)
```

gramian 矩阵为

```
g'
ans =
 1.0062e-01 6.8039e-05 1.0055e-05
```

gramian 矩阵表明后面 2 个状态可以删除

```
sysr = modred(sysb,[2 3],'del')
```

原系统的一阶近似为

```
zpk(sysr)
Zero/pole/gain:
   1.0001
--------------
  (s+4.97)
```

将原系统与降阶系统进行比较，见图 2.3.1。

```
bode(sys,'-',sysr,'x')
```

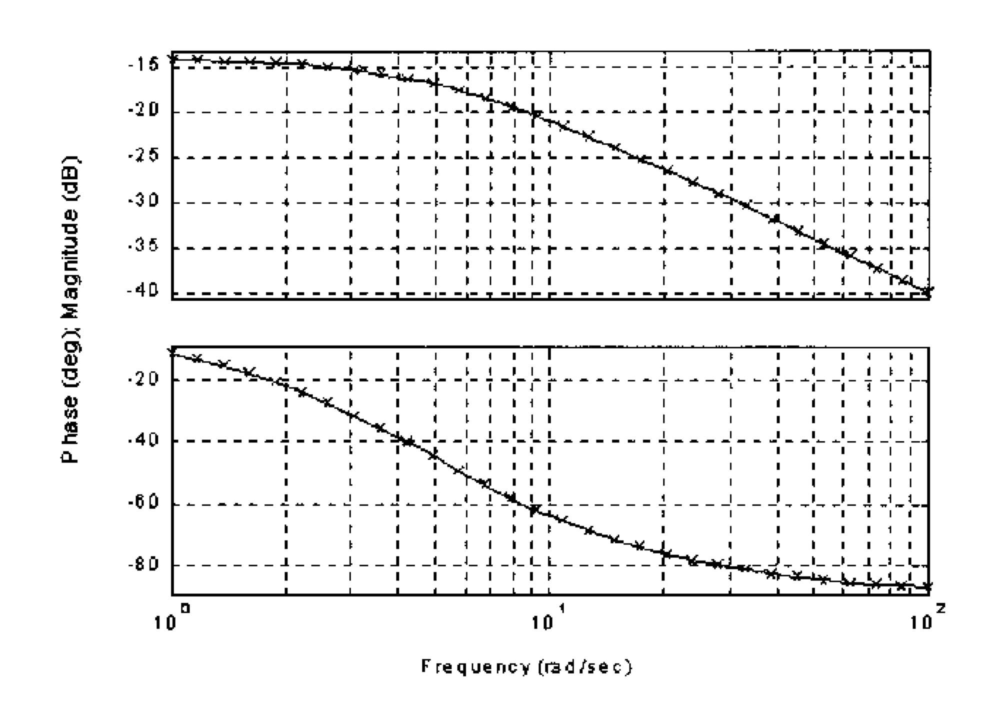


图 2.3.1　原系统与降阶系统 bode 图比较

10　minreal

功能：状态空间的最小实现。

格式：sysr = minreal(sys)

sysr = minreal(sys,tol)

说明：sysr = minreal(sys) 实现 LTI 对象 sys 的最小实现，即删除状态空间模型中的不可观和不可控状态；或者对消零极点模型中相同的零极点对。输出系统 sysr 具有最小的阶。

sysr = minreal(sys,tol)定义状态删除或者零极点对消中的容许误差 tol。大的容许误差将导致更多的状态删除或者零极点对消。缺省时 tol = sqrt(eps)。

举例：

```
g = zpk([],1,1)
h = tf([2 1],[1 0])
cloop = inv(1+g*h) * g
```

Matlab 返回(可见为非最小实现)

```
Zero/pole/gain:
     s (s-1)
---------------------
 (s-1) (s^2 + s + 1)
cloop = minreal(cloop)
```

Matlab 返回

```
Zero/pole/gain:
             s
       --------------
       (s^2 + s + 1)
```

11 modred

功能: 模型降阶。

格式: rsys = modred(sys,elim)

rsys = modred(sys,elim,'mdc')

rsys = modred(sys,elim,'del')

说明: modred 用于降低系统的维数。该函数提供了两种降阶方法，常与函数 balreal 连用。

rsys = modred(sys,elim)和 rsys = modred(sys,elim,'mdc')对 LTI 对象 sys 进行降阶。索引向量 elim 给出需要删除的状态的索引。降阶后的系统 rsys 具有与原系统相匹配的直流增益(DC gain)。

rsys = modred(sys,elim,'del')仅仅简单删除要删除的状态。该调用不能保证匹配的直流增益。但在频域降阶后的系统 rsys 具有与原系统更好的近似。

举例: 系统传递函数为

$$H(s)=\frac{s^3+11s^2+36s+26}{s^4+14.6s^3+74.96s^2+153.7s+99.65}$$

降阶前要首先进行降阶实现

```
h = tf([1 11 36 26],[1 14.6 74.96 153.7 99.65])
[hb,g] = balreal(h)
g'
ans =
1.3938e-01 9.5482e-03 6.2712e-04 7.3245e-06
```

gramians 后面的三个元素很小，因此可以将其消除，同时使用两种方法以进行比较。

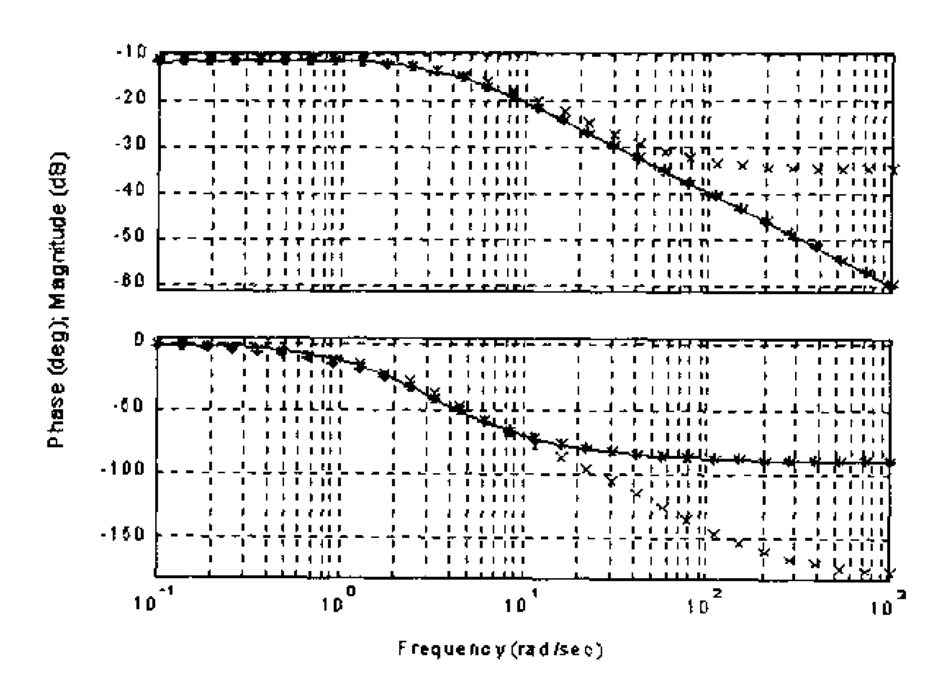


图 2.3.2 降阶系统与原系统 bode 图比较

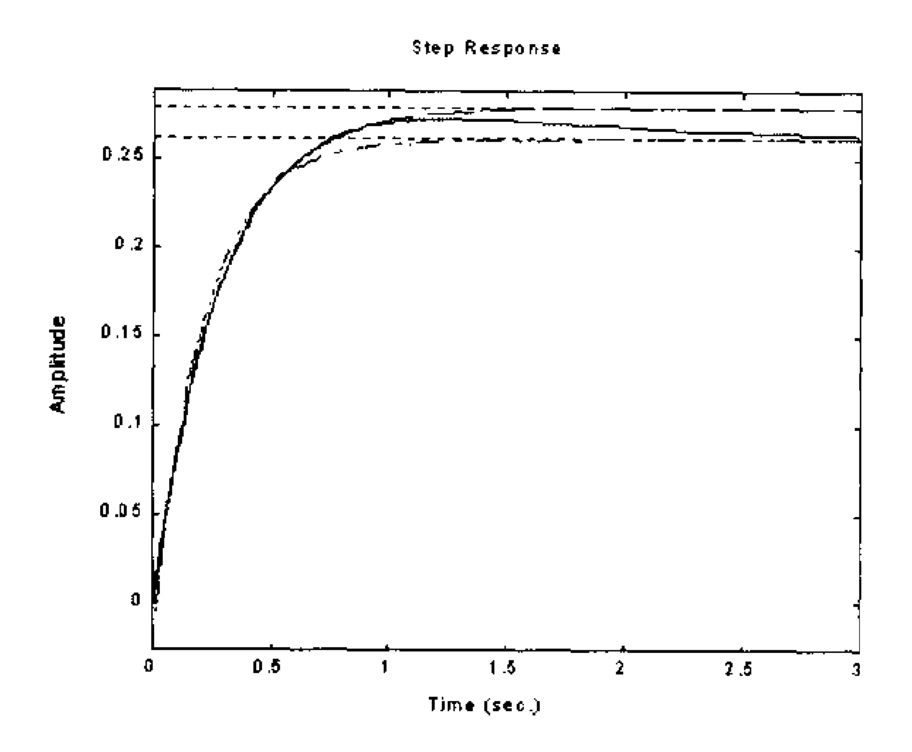


图 2.3.3 降阶系统与原系统阶跃响应比较

```
hmdc = modred(hb,2:4,'mdc')
hdel = modred(hb,2:4,'del')
```

伯特图比较(见图 2.3.2)。

```
bode(h,'-',hmdc,'x',hdel,'*')
```

阶跃响应比较(见图 2.3.3)。

```
step(h,'-',hmdc,'-.',hdel,'--')
```

2.4 系统特性函数

对于系统的其他一些简单的特性，如系统的零极点、噪声响应特性以及直流增益、自然频率、阻尼系数等。控制系统工具箱也提供了相应的函数来进行计算，见表 2.4.1。要注意的是，对于系统的零极点，通过模型转换函数 pzk 也可以获得。

表 2.4.1 系统特性函数列表

函数名称	功能
damp	计算系统的自然频率和衰减因子
dcgain	计算系统的直流(D.C.)增益
covar	计算白噪声驱动的系统协方差矩阵
dsort	按幅值对离散时间系统极点排序
esort	按实部对连续时间系统极点排序
pole, eig	计算系统极点
pzmap	系统零极点绘制
tzero	计算系统传输零点

1 damp

功能：计算系统的自然频率和衰减因子。

格式：[Wn,Z] = damp(sys)

[Wn,Z,P] = damp(sys)

说明：damp 函数用于计算自然频率和衰减因子。当不带输出变量时，在屏幕上列表显示特征值表、衰减比率及自然频率。

[Wn,Z] = damp(sys)计算 LTI 对象 sys 的自然频率 Wn 和衰减因子 Z。Wn 和 Z 为向量。对于离散时间系统，若其极点为 z，采样周期为 Ts，则通过计算等效的连续时间系统极点来获得。如果采样周期未定义，则返回值为空。

[Wn,Z,P] = damp(sys)同时返回系统的极点向量 P。

举例：计算并显示下述系统的极点、自然频率和衰减因子

$$H(s)=\frac{2s^2+5s+1}{s^2+2s+3}$$

```
H = tf([2 5 1],[1 2 3])
```

Matlab 返回

```
Transfer function:
```

```
  2 s^2 + 5 s + 1
------------------
  s^2 + 2 s + 3
damp(H)
```

Matlab 返回

```
Eigenvalue                    Damping      Freq. (rad/s)
-1.00e+000 + 1.41e+000i  5.77e-001    1.73e+000
-1.00e+000 -11.41e+000i  5.77e-001    1.73e+000
```

2 dcgain

功能： 计算系统的直流(D.C.)增益。

格式： k = dcgain(sys)

说明： dcgain 函数用于计算系统的直流增益。对于由(*A*,*B*,*C*,*D*)确定的连续时间系统，直流增益可通过下式计算获得：

$$K = D - CA^{-1}B$$

对于由(*A*,*B*,*C*,*D*)确定的离散时间系统，直流增益可通过下式计算获得：

$$K = D + C(I - A)^{-1}B$$

k = dcgain(sys) 计算 LTI 对象 sys 的直流(D.C.)增益。计算方法如上所述。

举例： 计算下述 MIMO 系统的直流增益

$$H(s)=\begin{bmatrix} 1 & \dfrac{s-1}{s^2+s+3} \\ \dfrac{1}{s+1} & \dfrac{s+2}{s-3} \end{bmatrix}$$

```
H = [1 tf([1 -1],[1 1 3]) ; tf(1,[1 1]) tf([1 2],[1 -3])]
dcgain(H)
```

Matlab 返回

```
ans =
1.0000 -0.3333
1.0000 -0.6667
```

3 covar

功能： 计算白噪声驱动的系统状态和输出协方差矩阵。

格式： [P,Q] = covar(sys,W)

说明： covar 函数用于计算白噪声驱动的系统状态和稳态输出协方差矩阵。当输入为高斯白噪声 w 且有：

$$E\left(w(t)w(\tau)^T\right) = W\delta(t-\tau) \quad \text{(连续时间系统)}$$

$$E\left(w(k)w(i)^T\right) = W\delta_{ki} \quad \text{(离散时间系统)}$$

稳态输出和系统状态的协方差矩阵定义为：

$$P=E\left(yy^{T}\right) \qquad Q=E\left(xx^{T}\right)$$

[P,Q] = covar(sys,W)计算 LTI 对象 sys 的系统状态和稳态输出协方差矩阵。其中 P，Q，W 如上定义。

4 dsort

功能： 按幅值对离散时间系统极点排序。

格式： s = dsort(p)

[s,ndx] = dsort(p)

说明： s = dsort(p)对输入的离散时间系统的极点向量 p 依幅值的降序进行排序，返回值为排序结果。因此不稳定极点在前面。

[s,ndx] = dsort(p)同时返回索引向量 ndx。

5 esort

功能： 按实部对连续时间系统极点排序。

格式： s = esort(p)

[s,ndx] = esort(p)

说明： s = esort(p) 对输入的连续时间系统的极点向量 p 依实部的降序进行排序，返回值为排序结果。因此不稳定极点在前面。

[s,ndx] = esort(p) 同时返回索引向量 ndx。

6 pole, eig

功能： 计算系统极点。

格式： p = pole(sys)

poles = eig(sys)

说明： p = pole(sys)和 poles = eig(sys)计算 LTI 对象 sys 的极点。对于状态空间模型，系统极点为矩阵 A 的特征值；对于传递函数模型，系统极点为分母多项式的根；对于 MIMO 系统，系统极点为每个传递函数分量极点的子集。由于数值

△计算引入误差，该函数不能保证多重极点的准确计算。

7 pzmap

功能： 系统零极点绘制。

格式： pzmap(sys)

[p,z] = pzmap(sys)

说明： pzmap(sys)绘制 LTI 对象 sys 的零极点图。对于 SISO 系统， 该函数绘制传递函数的零极点；对于 MIMO 系统，该函数绘制系统的极点和传输零点(transmission zeros)。绘制时极点以‘x’表示，零点以‘o’表示。

[p,z] = pzmap(sys)返回 LTI 对象 sys 的极点向量 p 和零点向量 z。不进行绘制。

举例： 绘制如下系统的零极点图，见图 2.4.1。

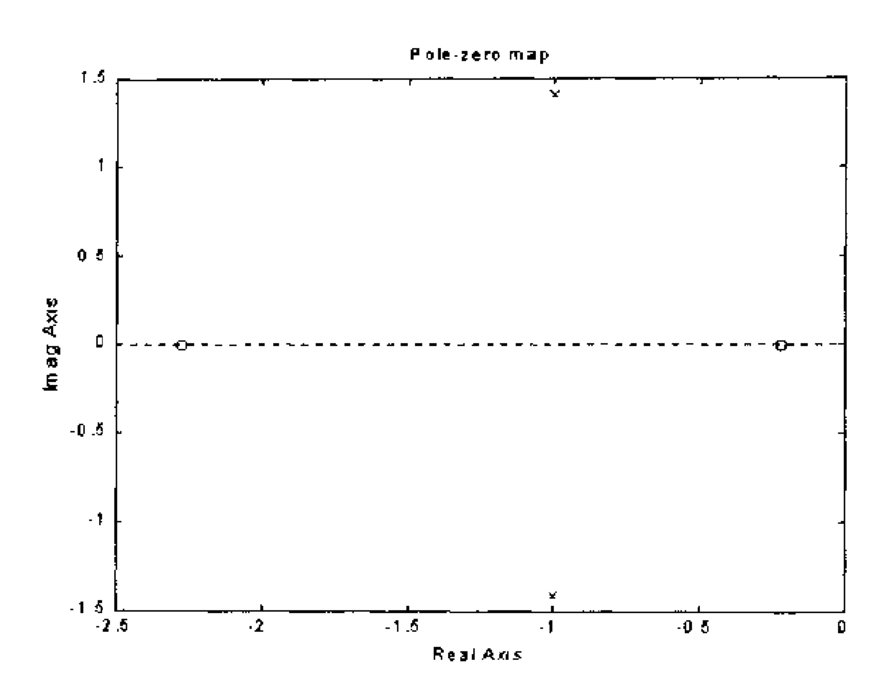


图 2.4.1 系统的零极点图

$$H(s)=\frac{2s^2+5s+1}{s^2+2s+3}$$

```
H = tf([2 5 1],[1 2 3])
pzmap(H)
```

8 tzero

功能：计算系统传输零点(transmission zeros)。

格式：z = tzero(sys)

[z,gain] = tzero(sys)

z = tzero(a,b,c,d)

说明：tzero 函数用于计算 SISO 系统的零点以及 MIMO 系统的传输零点。

z = tzero(sys) 算 LTI 对象 sys 的零点 z。其中，z 以列向量返回。

[z,gain] = tzero(sys)如果系统为 SISO 系统，则同时返回系统的增益。

z = tzero(a,b,c,d)直接以状态空间矩阵计算。它等价于 z = tzero(ss(a,b,c,d))。

2.5 系统根轨迹

根轨迹法是一种求解闭环特征方程根的简便的图解方法，它根据系统的开环传递函数极点和零点的分布。依据一些简单的规则，研究开环系统某一个参数(常选作开环系统的增益k)从零到无穷大时，闭环系统极点在 S 平面上的轨迹。利用根轨迹法能够分析结构和参数确定的系统的稳定性及系统的动态响应特性。还可根据对系统动态和静态特性的要求确定可变参数，调整开环零极点的位置甚至数目。因此在控制系统的分析和设计中根轨迹是一种很实

用的工程方法。

控制系统工具箱对根轨迹的函数支持见表 2.5.1。此外，工具箱还提供了一个根轨迹的 GUI 分析工具，见 2.10 节的详细说明。

表 2.5.1 系统根轨迹绘制及分析函数列表

函数名称	功能
rlocfind	计算给定根的根轨迹增益
rlocus	求系统根轨迹
sgrid	绘制连续时间系统根轨迹和零极点图中的阻尼系数和自然频率栅格
zgrid	绘制离散时间系统根轨迹和零极点图中的绘制阻尼系数和自然频率栅格

1 rlocfind

功能：计算给定根的根轨迹增益。

格式：[k,poles] = rlocfind(sys)

[k,poles] = rlocfind(sys,p)

说明：rlocfind 函数可计算出与根轨迹上极点相对应的根轨迹增益。该函数既适用于连续时间系统，也适用于离散时间系统。

[k,poles] = rlocfind(sys) 在 LTI 对象 sys 的根轨迹图中显示出十字光标，当用户选择其中一点时，其相应的增益由 k 记录，与增益相关的所有极点记录在 poles 中。若要使用该函数，必须首先在当前窗口绘制系统的根轨迹。

[k,poles] = rlocfind(sys,p)定义要得到增益的根矢量 p。

2 rlocus

功能：求系统根轨迹。

格式：rlocus(sys)

rlocus(sys,k)

[r,k] = rlocus(sys)

r = rlocus(sys,k)

说明：rlocus 函数用于计算 SISO 系统的 Evans(伊文斯)根轨迹。根轨迹可用于研究改变反馈增益对系统极点分布的影响，从而提供系统时域和频域响应的分析。设系统传递函数 $g(s)$，反馈 $k*f(s)$的受控系统，则其闭环传递函数为

$$H(s)=\frac{g(s)}{1+kg(s)f(s)}$$

当系统调用不含输出变量时，rlocus 在当前图形窗口中绘制出系统的根轨迹图。该函数既适用于连续时间系统，也适用于离散时间系统。

rlocus(sys)计算 SISO 开环 LTI 对象 sys 的根轨迹。如上所述，绘制 1+kg(s)f(s) 的根轨迹，增益 k 将自动选取。

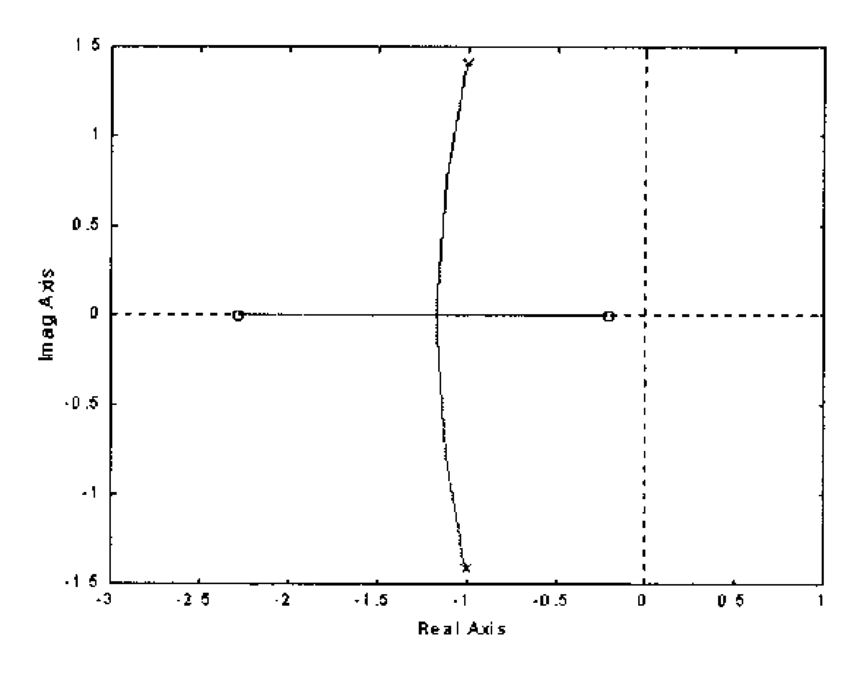


图 2.5.1 传递函数的根轨迹

rlocus(sys,k)显式设置增益 k。

[r,k] = rlocus(sys)返回系统计算的增益 k 和闭环系统对应的极点，即对应与 k(i) 处的增益，系统极点为 r(i)。

r = rlocus(sys,k) 显式设置增益 k。

举例：绘制下述开环传递函数的根轨迹，见图 2.5.1。

$$G(s)=\frac{2s^2+5s+1}{s^2+2s+3}$$

```
h = tf([2 5 1],[1 2 3]);
rlocus(h)
```

3 sgrid

功能：绘制连续时间系统根轨迹和零极点图中的阻尼系数和自然频率栅格。

格式：sgrid

sgrid(z,wn)

说明：sgrid 在连续系统的根轨迹或零极点图上绘制栅格线。栅格线由等阻尼系数和等自然频率线构成，阻尼系数步长为 0.1，范围从 0 到 1。自然频率步长为 1 弧度/秒，范围从 0 到 10。在绘制前，当前窗口必须包含连续时间系统的 s 平面根轨迹或者零极点图。

sgrid(z,wn)显式定义想要绘制的阻尼系数向量 z 和自然频率向量 wn。

举例：仍以 rlocus 函数中的系统为例

```
H = tf([2 5 1],[1 2 3])
```

Matlab 将返回

```
Transfer function:
```

```
  2 s^2 + 5 s + 1
-------------------------
  s^2 + 2 s + 3
```

rlocus(H) 见图 2.5.1。

sgrid 见图 2.5.2。

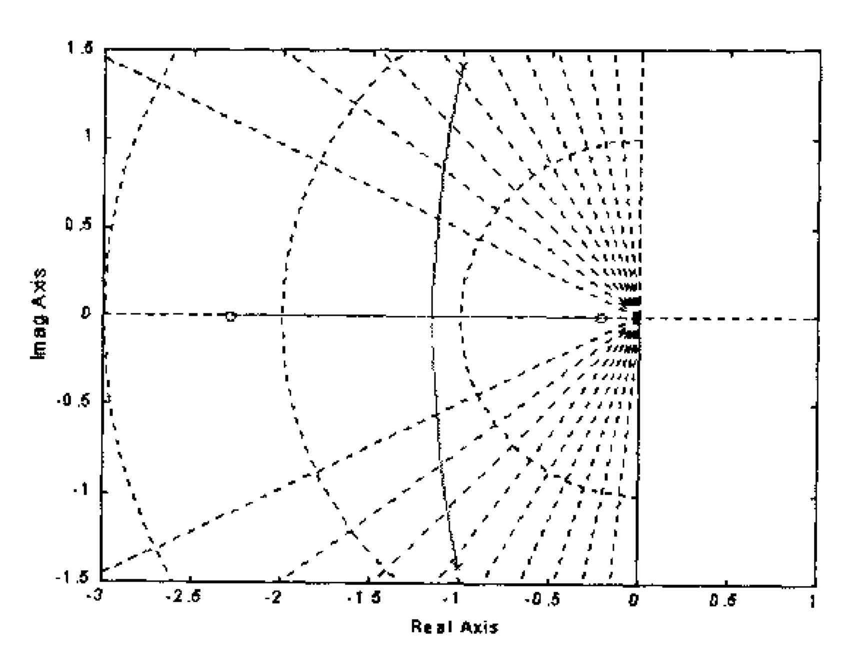


图 2.5.2 根轨迹自然频率栅格

4 zgrid

功能：绘制离散时间系统根轨迹和零极点图中的阻尼系数和自然频率栅格。

格式：zgrid

zgrid(z,wn)

说明：zgrid 在离散系统的根轨迹图或零极点图上绘制出栅格线。栅格线由等阻尼系数和自然频率线构成，阻尼系数步长为 0.1，范围从 0 到 1，自然频率步长为 π/10，从 0 到 π。在绘制前，当前窗口必须包含离散时间系统的 z 平面根轨迹或者零极点图。

zgrid(z,wn) 显式定义想要绘制的阻尼系数向量 z 和自然频率向量 wn。

举例：对下述系统进行 zgrid 绘制，见图 2.5.3。

$$H(z)=\frac{2z^2-3.4z+1.5}{z^2-1.6z+0.8}$$

```
H = tf([2 -3.4 1.5],[1 -1.6 0.8],-1)
```

Matlab 将返回：

```
Transfer function:
2 z^2 -3.4 z + 1.5
-------------------
```

```
z^2 -1.6 z + 0.8
Sampling time: unspecified
```

输入

```
rlocus(H)
zgrid
axis('square')
```

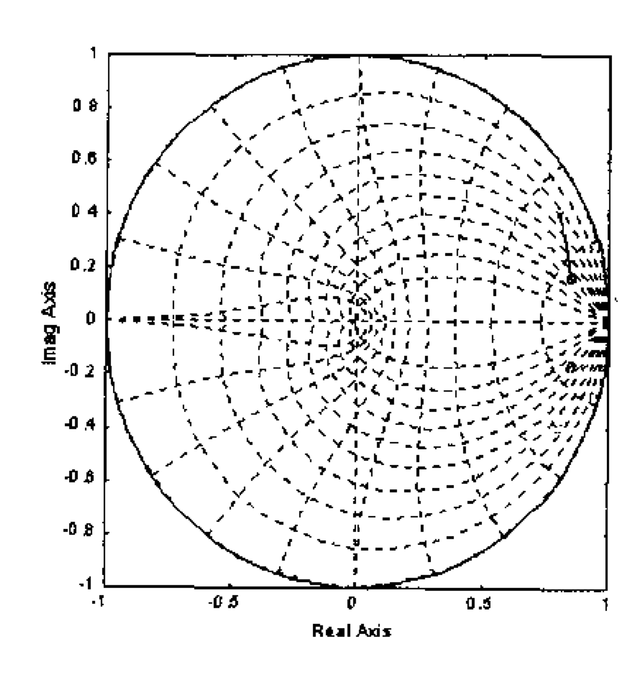


图 2.5.3　离散系统阻尼系数和自然频率

2.6　系统频率响应

2.6.1　频率响应计算

1　频率响应的计算

设 LTI 系统的传递函数由式(2.1.2)给出，则系统的频率响应为

$$G(jw)=\frac{b_1(jw)^m+b_2(jw)^{m-1}+\cdots+b_m(jw)+b_{m+1}}{a_1(jw)^n+a_2(jw)^{n-1}+\cdots+a_n(jw)+a_{n+1}}$$

若系统由式(2.1.5)的状态空间模型来描述，则系统的频率响应为

$$G(jw)=C[jwI-A]^{-1}B+D$$

2 频率响应曲线绘制

通过频率响应曲线对系统进行频域分析是系统分析常用的方法。常用的频率响应曲线绘制包括 Bode 图(幅频/相频特性曲线)、Nyquist 曲线，以及 Nichols 图等。

- Bode 图

Bode 图实际上是系统的对数频率特性图，即对于系统的频率响应函数 $G(jw)$，取

$$u = \lg w$$
$$L(G(jw)) = \lg|G(jw)|$$
$$\theta(G(jw)) = \arg(G(jw))$$

则 $L(G(jw))$为系统的对数幅频特性函数；$\theta(G(jw))$为系统的对数相频特性函数；总称为对数频率特性函数。

- Nyquist 图

Nyquist 图为系统的极坐标频率特性图，它根据系统的频率响应函数 $G(jw)$在复平面上绘制系统的幅相特性。

根据系统的开环 Nyquist 曲线，可以判断系统的稳定性。

Nyquist 稳定判据：若系统的开环传递函数在右半 s 平面有 p 个极点，则闭环系统稳定的充要条件是：当 w 从 $-\infty$变到$+\infty$时，Nyquist 曲线延逆时针方向包围复平面上 -1 点 p 圈。其中，p 为系统开环传递函数在右半 s 平面的极点数。

- 稳定裕度

设系统是稳定的，但非条件稳定，且其开环频率特性曲线不包围-1 点。把开环幅频特性函数的值等于 1 的角频率记作 w_c，则把这一角频率上开环相频特性函数值大于 $-180°$ 的值称为相角稳定裕度，记作 γ，即

$$A(G(jw_c)) = 1$$
$$\gamma = \theta(G(jw_c)) - (-180^\circ)$$

同时把开环相频特性函数的值等于 $-180°$ 的角频率记作 w_o，则把这一角频率上开环幅频特性函数值的导数称为增益稳定裕度，记作 Kg。Kg 用 dB 表示，即

$$\theta(G(jw_o)) = -180^\circ$$
$$Kg = -1/A(G(jw_o))$$

2.6.2 频率响应绘制函数

控制系统工具箱中关于系统频域曲线绘制及频域分析的函数见表 2.6.1。

表 2.6.1　系统频率响应绘制及分析函数列表

函数名称	功　能
bode	Bode 图绘制
evalfr	计算系统单频率点的频率响应
freqresp	计算系统频率响应
margin	计算系统的增益和相角稳定裕度
ngrid	Nichols 网格线绘制
nichols	Nichols 绘制
nyquist	Nyquist 绘制
sigma	系统的奇异值 Bode 图绘制

1　bode

功能：Bode 图绘制。

格式：
```
bode(sys)
bode(sys,w)
bode(sys1,sys2,...,sysN)
bode(sys1,sys2,...,sysN,w)
bode(sys1,'PlotStyle1',...,sysN,'PlotStyleN')
[mag,phase,w] = bode(sys)
```

说明：Bode 函数计算并显示 LTI 系统的 Bode 图。当调用无输出变量时，bode 在当前图形窗口中进行 bode 图绘制。

bode(sys)计算并在当前窗口绘制 LTI 对象 sys 的 bode 图，可用于 SISO 或者 MIMO 的连续时间系统或者离散时间系统。绘制时的频率范围将依据系统的零极点决定。

bode(sys,w)显式定义绘制时的频率范围或者频率点 w。若要定义频率范围，w 必须具有[wmin,wmax]格式；如果定义频率点，则 w 必须为由需要频率点频率组成的向量。

bode(sys1,sys2,...,sysN)和 bode(sys1,sys2,...,sysN,w)同时在一个窗口绘制多个 LTI 对象的 Bode 图。这些系统必须具有同样多的输入和输出数，但可以同时含有离散时间和连续时间系统。该调用常用于多个系统 bode 图的比较。

bode(sys1,'PlotStyle1',...,sysN,'PlotStyleN') 定义每个绘制的绘制属性。其中 PlotStyle1 和 PlotStyleN 为 Matlab 标准命令 plot 支持的各种属性标识字符串。

[mag,phase,w] = bode(sys)和[mag,phase] = bode(sys,w) 计算 bode 图数据，且不在窗口上显示。其中，mag 为 bode 图的幅度值，phase 为 bode 图的相位值，w 为 bode 图的频率点。mag 和 phase 均为 3 维向量，其大小为：(number of outputs)*(number of inputs)*(length of w)，对于 SISO 系统，有

$$\omega_k = W(k)$$
$$mag(1,1,k) = |h(j\omega)|$$
$$phase(1,1,k) = \angle h(j\omega)$$

对于 MIMO 系统，第 i 个输入与第 j 个输出的传递函数 h_{ij}，有

$$mag(i,j,k)=\left|h_{ij}(j\omega)\right|$$
$$phase(i,j,k)=\angle h_{ij}(j\omega)$$

举例：绘制下述系统的 bode 图，见图 2.6.1。

$$H(s)=\frac{s^2+0.1s+7.5}{s^4+0.12s^3+9s^2}$$

```
g = tf([1 0.1 7.5], [1 0.12 9 0 0])
bode(g)
```

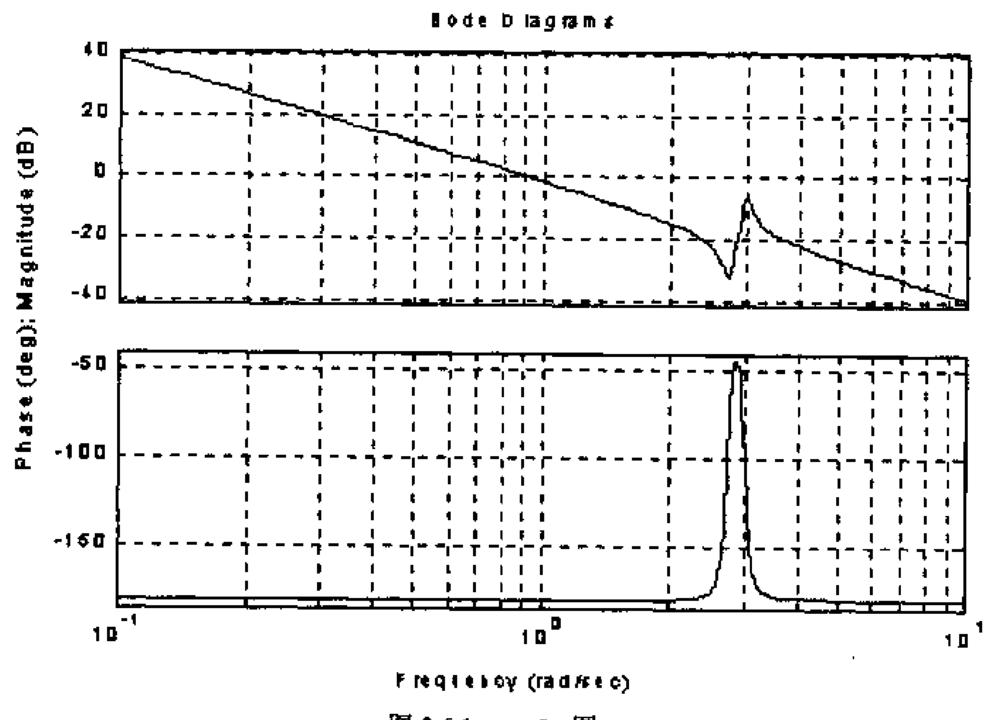


图 2.6.1 Bode 图

2 evalfr

功能： 计算系统单频率点的频率响应。

格式： frsp = evalfr(sys,f)

说明： evalfr 用于计算系统单个复数频率点的频率响应。其计算方法为

$$H(f)=D+C(fI-A)^{-1}B$$

frsp = evalfr(sys,f)计算传递函数 LTI 对象 sys 在频率点 f 处的频率响应。其中，f 要以复数给出。

举例： 计算下述离散时间系统在 1+j 处的频率响应

$$H(z)=\frac{z-1}{z^2+z+1}$$

```
H = tf([1 -1],[1 1 1],-1)
z = 1+j
evalfr(H,z)
```

Matlab 将返回

```
ans =
2.3077e-01 + 1.5385e-01
```

3 freqresp

功能：计算系统频率响应。

格式：H = freqresp(sys,w)

说明：freqresp 函数用于计算系统在实数频率处的频率响应。对于以状态空间描述的连续时间系统，其计算方法为

$$H(j\omega)=D+C(j\omega I-A)^{-1}B$$

对于以传递函数描述的离散时间系统，首先对频率 w 进行以下转换

$$z=e^{j\omega T_s}$$

其中，T_s 为采样周期，然后带入传递函数进行计算。

H = freqresp(sys,w) 计算 LTI 对象 sys 在实数频率点 w 处的频率响应 H。其中 w 为频率点向量，H 为 3 维向量，其大小为：(number of outputs)*(number of inputs)*(length of w)。对于 SISO 系统，H(1,1,k)给出系统在频率 w(k)处的频率响应；对于 MIMO 系统 H(I,j,k)给出了第 i 个输入与第 j 个输出间在频率 w(k)处的频率响应。

举例：计算下述系统在频率 1，10，100 处的频率响应

$$H(s)=\begin{bmatrix} 0 & \dfrac{1}{s+1} \\ \dfrac{s-1}{s+2} & 1 \end{bmatrix}$$

```
w = [1 10 100]
H = freqresp(P,w)
```

Matlab 将返回

```
H(:,:,1) =
0 0.5000-0.5000i
-0.2000+ 0.6000i 1.0000
H(:,:,2) =
0 0.0099-0.0990i
0.9423+ 0.2885i 1.0000
H(:,:,3) =
0 0.0001-0.0100i
```

```
0.9994+ 0.0300i 1.0000
```

4 margin

功能： 计算系统的增益和相位裕度。

格式： [Gm,Pm,Wcg,Wcp] = margin(sys)

[Gm,Pm,Wcg,Wcp] = margin(mag,phase,w)

margin(sys)

说明： margin 函数可从频率响应数据中计算出增益、相位裕度以及相应的交叉频率。增益和相位裕度是针对开环 SISO 系统而言的，它指示出当系统闭环时的相对稳定性。当不带输出变量引用时，margin 可在当前图形窗口中绘制出裕度的 Bode 图。

[Gm,Pm,Wcg,Wcp] = margin(sys) 计算 LTI 对象 sys 的增益和相位裕度。返回值中，Gm 对应于系统的增益裕度，Wcg 为其响应的交叉频率；Pm 对应于系统的相位裕度，Wcp 为其响应的交叉频率。

[Gm,Pm,Wcg,Wcp] = margin(mag,phase,w)根据由 mag、phase 和 w 给出的系统 bode 图数据计算系统的增益和相位裕度。其中 mag 给出 bode 图数据的幅值数据，phase 给出 bode 图数据的相位数据，w 为频率向量。

margin(sys) 在当前图形窗口中绘制出系统裕度的 Bode 图。

举例： 计算下述离散时间系统的增益和相位裕度，并绘制出系统裕度的 Bode 图，见图 2.6.2。

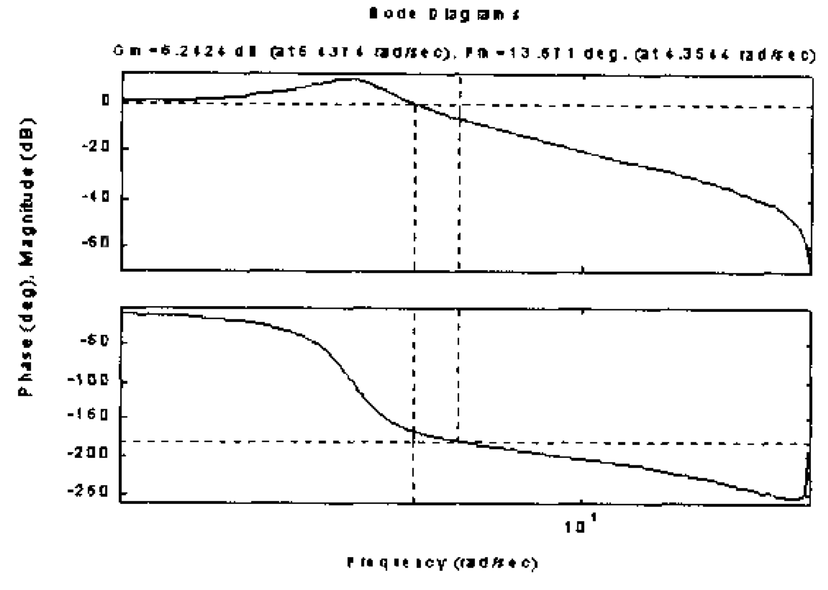


图 2.6.2 系统稳定裕度的 Bode 图

```
hd = tf([0.04798 0.0464],[1 -1.81 0.9048],0.1)
```

Matlab 将返回

```
Transfer function:
     0.04798 z + 0.0464
```

```
-------------------------
        z^2 -1.81 z + 0.9048
Sampling time: 0.1
   [Gm,Pm,Wcg,Wcp] = margin(hd);
   [Gm,Pm,Wcg,Wcp]
```

Matlab 将返回

```
   ans =
      2.0517 13.5712 5.4374 4.3544
```

5　ngrid

功能： Nichols 网格线绘制。

格式： ngrid

说明： ngrid 函数可给 Nichols 曲线图加上网格线，Nichols 曲线将 h/(1+h)与 h 相关联，当 h 为 SISO 系统的开环频率响应时，则 h/(1+h)为系统的单位闭环负反馈的频率响应。

ngrid 绘制 Nichols 网格线，每条网格线具有同样的幅值和相位，幅值取 –40dB~40dB，相位取 –360° ~0° 。

举例： 绘制下述系统的 Nichols 曲线和网格线，见图 2.6.3。

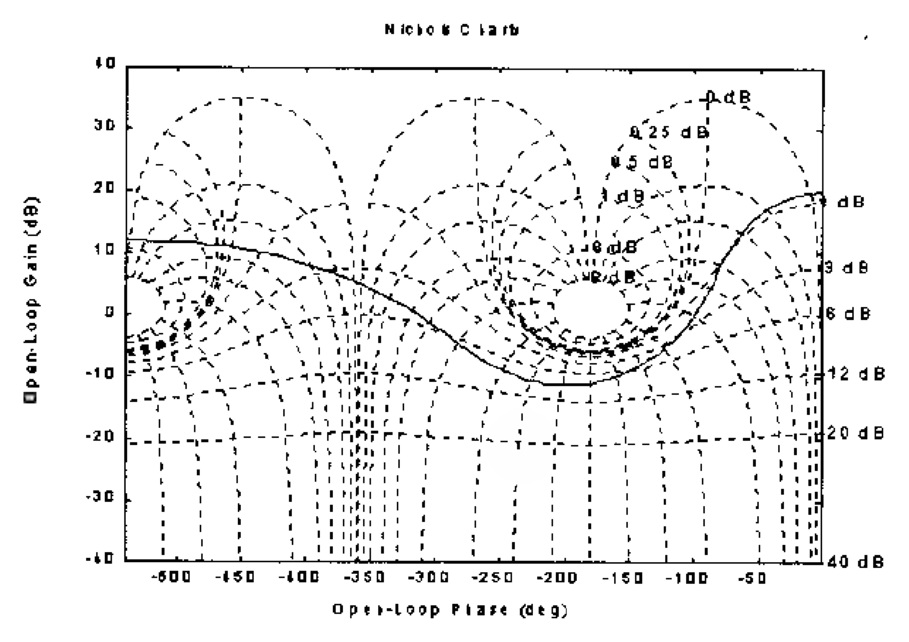


图 2.6.3　系统的 Nichols 曲线和网格

$$H(s)=\frac{-4s^4+48s^3-18s^2+250s+600}{s^4+30s^3+282s^2+525s+60}$$

```
H = tf([-4 48 -18 250 600],[1 30 282 525 60])
```

Matlab 将返回

```
Transfer function:
```

```
   -4 s^4 + 48 s^3 -18 s^2 + 250 s + 600
  ----------------------------------------
    s^4 + 30 s^3 + 282 s^2 + 525 s + 60
nichols(H)
ngrid
```

6 nichols

功能：Nichols 绘制。

格式：nichols(sys)

nichols(sys,w)

nichols(sys1,sys2,...,sysN)

nichols(sys1,sys2,...,sysN,w)

nichols(sys1,'PlotStyle1',...,sysN,'PlotStyleN')

[mag,phase,w] = nichols(sys)

说明：nichols 函数可绘制 LTI 系统的 Nichols 频率响应曲线，Nichols 曲线能够分析开环和闭环系统的特性。当调用无输出变量时，函数将在当前图形窗口中直接绘制系统的 Nichols 曲线。

nichols(sys)在当前窗口绘制 LTI 对象 sys 的 nichols 曲线，可应用于 SISO 或者 MIMO 的连续时间系统或者离散时间系统。当系统为 MIMO 时，该函数产生一组 nichols 曲线，每个输入输出通道对应一个。绘制时的频率范围将依据系统的零极点决定。

nichols(sys,w) 显式定义绘制时的频率范围或者频率点 w。要定义频率范围，w 必须具有[wmin,wmax]格式；如果定义频率点，则 w 必须为由需要频率点频率组成的向量。

nichols(sys1,sys2,...,sysN)和 nichols(sys1,sys2,...,sysN,w) 同时在一个窗口绘制多个 LTI 对象的 nichols 图。这些系统必须具有同样多的输入和输出数，但可以同时含有离散时间和连续时间系统。该调用常用于多个系统 nichols 图的比较。

nichols(sys1,'PlotStyle1',...,sysN,'PlotStyleN') 定义每个绘制的绘制属性。其中 PlotStyle1 和 PlotStyleN 为 Matlab 标准命令 plot 支持的各种属性标识字符串。

[mag,phase,w] = nichols(sys)和[mag,phase] = nichols(sys,w) 计算 nichols 图数据，且不在窗口上显示。其中，mag 为 nichols 图的幅度值，phase 为 nichols 图的相位值，w 为 nichols 图的频率点。mag 和 phase 均为 3 维向量，其大小为：(number of outputs)*(number of inputs)*(length of w)，其意义与 bode 函数相同，详见 bode 函数。

7 nyquist

功能：求系统的 Nyquist(奈奎斯特)频率曲线。

格式：nyquist(sys)

nyquist(sys,w)

nyquist(sys1,sys2,...,sysN)

nyquist(sys1,sys2,...,sysN,w)

nyquist(sys1,'PlotStyle1',...,sysN,'PlotStyleN')

[re,im,w] = nyquist(sys)

说明：nyquist 函数可计算 LTI 系统的 Nyquist 频率曲线。Nyquist 曲线用来分析包括增益裕度、相位裕度及稳定性在内的系统特性。当调用无输出变量时，nyquist 函数会在当前图形窗口中直接绘制出 Nyquist 曲线。

nyquist(sys)在当前窗口绘制 LTI 对象 sys 的 nyquist 曲线。可应用于 SISO 或者 MIMO 的连续时间系统或者离散时间系统。当系统为 MIMO 时，该函数产生一组 nyquist 曲线，每个输入输出通道对应一个。绘制时的频率范围将依据系统的零极点决定。

nyquist(sys,w) 显式定义绘制时的频率范围或者频率点 w。若要定义频率范围，w 必须具有[wmin,wmax]格式；如果定义频率点，则 w 必须为由需要频率点频率组成的向量。

nyquist(sys1,sys2,...,sysN)和 nyquist(sys1,sys2,...,sysN,w) 同时在一个窗口绘制多个 LTI 对象的 nyquist 图。这些系统必须具有同样多的输入和输出数，但可以同时含有离散时间和连续时间系统。该调用常用于多个系统 nyquist 图的比较。

nyquist(sys1,'PlotStyle1',...,sysN,'PlotStyleN') 定义每个绘制的绘制属性。其中 PlotStyle1 和 PlotStyleN 为 Matlab 标准命令 plot 支持的各种属性标识字符串。

[re,im,w] = nyquist(sys)和[re,im] = nyquist(sys,w)返回系统在频率 w 处的频率响应。其中，re 为频率响应的实部，im 为频率响应的虚部，w 为频率点。re 和 im 均为 3 维向量，其大小为：(number of outputs)*(number of inputs)*(length of w)，对于 SISO 系统，有

$$\omega_k = W(k)$$
$$re(1,1,k) = re(h(j\omega))$$
$$im(1,1,k) = im(h(j\omega))$$

对于 MIMO 系统，第 i 个输入与第 j 个输出的传递函数 h_{ij}，有

$$re(i,j,k) = re(h_{ij}(j\omega))$$
$$im(i,j,k) = im(h_{ij}(j\omega))$$

举例：绘制下述系统的 nyquist 曲线，见图 2.6.4。

$$H(s) = \frac{2s^2 + 5s + 1}{s^2 + 2s + 3}$$

```
H = tf([2 5 1],[1 2 3])
nyquist(H)
```

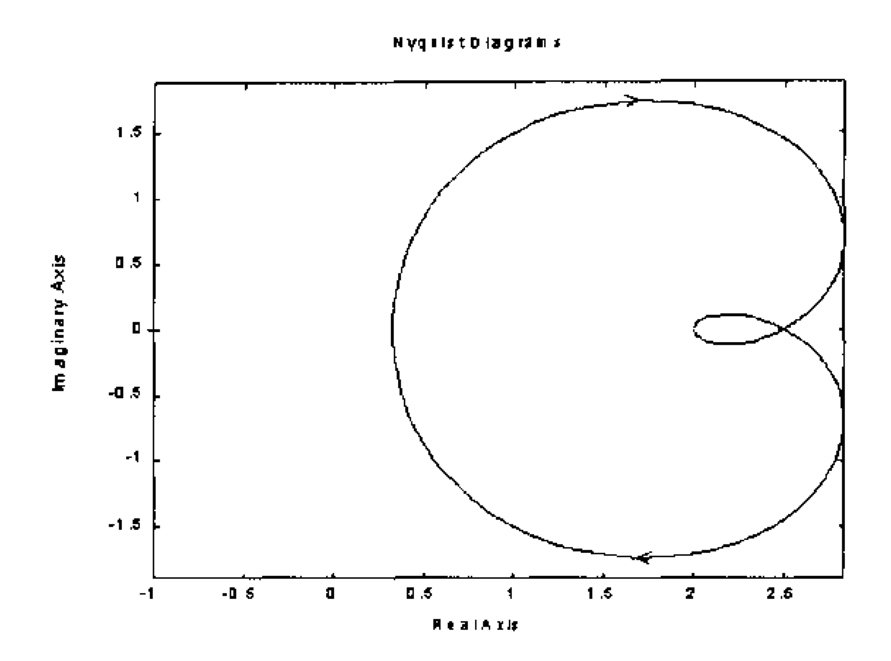


图 2.6.4　系统的 nyquist 曲线

8　sigma

功能：求系统的奇异值响应。

格式：sigma(sys)

sigma(sys,w)

sigma(sys,w,type)

sigma(sys1,sys2,...,sysN)

sigma(sys1,sys2,...,sysN,w)

sigma(sys1,sys2,...,sysN,w,type)

sigma(sys1,'PlotStyle1',...,sysN,'PlotStyleN')

[sv,w] = sigma(sys)

说明：sigma 函数可计算系统的奇异值响应。它是 Bode 幅值响应在 MIMO 系统的扩展，广泛应用于系统的鲁棒分析。当调用无输出变量时，sigma 函数在当前图形窗口中绘出奇异值 Bode 图。

sigma(sys) 在当前窗口绘制 LTI 对象 sys 的奇异值响应。可应用于 SISO 或者 MIMO 的连续时间系统或者离散时间系统。绘制时的频率范围将依据系统的零极点决定。

sigma(sys,w) 显式定义绘制时的频率范围或者频率点 w。若要定义频率范围，w 必须具有[wmin,wmax]格式。如果定义频率点，则 w 必须为由需要频率点频率组成的向量。频率的单位必须为弧度/秒。

sigma(sys,w,type) 在当前窗口绘制 LTI 对象 sys 的其他类型的奇异值响应。其中，type 可为下述值之一：

- type = 1——绘制 H^{-1} 的奇异值响应（这里 H 为系统传递函数）；

- type = 2——绘制 I+H 的奇异值响应（这里 I 为单位矩阵）；
- type = 3——绘制 $I+H^{-1}$ 的奇异值响应。

sigma (sys1,sys2,...,sysN) 和 sigma (sys1,sys2,...,sysN,w)和 sigma(sys1,sys2,..., sysN,w,type) 同时在一个窗口绘制多个 LTI 对象的奇异值响应。这些系统必须具有同样多的输入和输出数，但可以同时含有离散时间和连续时间系统。该调用常用于多个系统 nyquist 图的比较。

sigma(sys1,'PlotStyle1',...,sysN,'PlotStyleN') 定义每个绘制的绘制属性。其中 PlotStyle1 和 PlotStyleN 为 Matlab 标准命令 plot 支持的各种属性标识字符串。

[sv,w] = sigma(sys)和 sv = sigma(sys,w) 返回系统在频率 w 处的奇异值响应。对于一个具有 Nu 个输入和 Ny 个输出的系统，sv 的大小为 min(Nu,Ny)*(length of w)，在频率 w(k)处系统的奇异值为 sv(:,k)。

举例：绘制下述系统的 H 和 I+H 的奇异值响应，见图 2.6.5。

$$H(s)=\begin{bmatrix} 0 & \dfrac{3s}{s^2+s+10} \\ \dfrac{s+1}{s+5} & \dfrac{2}{s+6} \end{bmatrix}$$

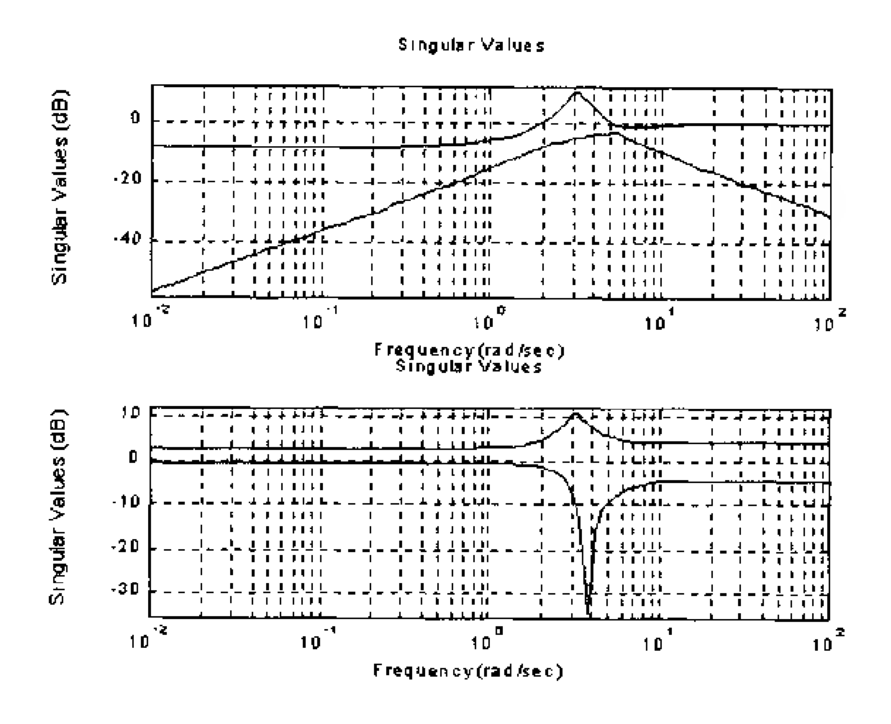


图 2.6.5　系统的 H 和 I+H 的奇异值响应

```
H = [0 tf([3 0],[1 1 10]) ; tf([1 1],[1 5]) tf(2,[1 6])]
subplot(211)
sigma(H)
subplot(212)
sigma(H,[ ],2)
```

2.7 系统时域响应

系统仿真实质上就是对描述系统的数字模型进行求解。对控制系统而言，其数学模型均可用微分方程或者差分方程给出。因此系统的仿真过程就表现为从给定的初始值出发，以某种数值算法逐步计算系统每个时刻的响应，从而绘制出系统的响应曲线并分析系统的性能。对于复杂系统的方针与分析，Matlab 由专门的 Simulink 来完成。关于 Simulink 的详细使用说明见有关书籍。

对于 LTI 系统，控制系统工具箱提供了若干个函数来完成对 LTI 系统的仿真，如系统的单位阶跃响应和系统的单位脉冲响应。它还提供一个生成任意信号及对 LTI 系统任意输入进行仿真的函数，见表 2.7.1。

表 2.7.1 系统时域分析函数列表

函数名称	功能
gensig	输入信号产生
impulse	计算系统脉冲响应
initial	计算系统零输入响应
lsim	对系统的任意输入进行仿真
step	计算系统阶跃响应

1 gensig

功能：输入信号产生。

格式：[u,t] = gensig(type,tau)

[u,t] = gensig(type,tau,Tf,Ts)

说明：[u,t] = gensig(type,tau)产生一个类型为 type 的信号序列 u(t)。信号周期为 tau。其中 type 可以为以下类型标识字符串之一：

- 'sin' ——正弦波；
- 'square' ——方波；
- 'pulse' ——脉冲序列。

[u,t] = gensig(type,tau,Tf,Ts)同时定义持续时间 Tf 和采样周期 Ts。

举例：生成一个周期为 5 秒，持续时间为 30 秒，采样周期为 0.1 秒的方波，见图 2.7.1。

```
[u,t] = gensig('square',5,30,0.1)
plot(t,u)
axis([0 30 -1 2])
```

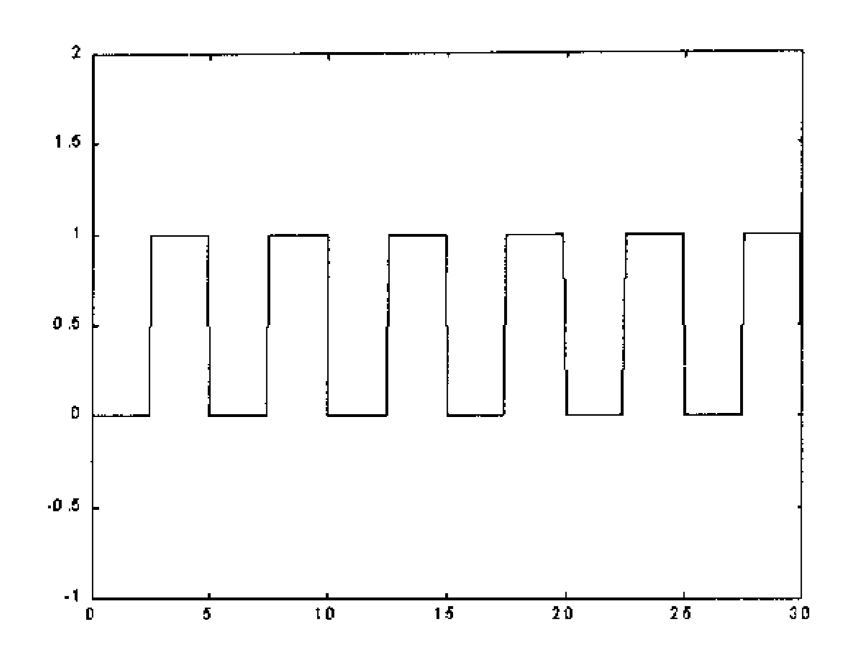


图 2.7.1　方波信号

2　impulse

功能： 计算系统脉冲响应。

格式：
```
impulse(sys)
impulse(sys,t)
impulse(sys1,sys2,...,sysN)
impulse(sys1,sys2,...,sysN,t)
impulse(sys1,'PlotStyle1',...,sysN,'PlotStyleN')
[y,t,x] = impulse(sys)
```

说明： impulse 函数用于计算系统的单位冲激响应。当调用无输出变量时，impulse 在当前图形窗口中直接绘出系统的单位冲激响应。

impulse(sys)计算并在当前窗口绘制 LTI 对象 sys 的脉冲响应，可用于 SISO 或者 MIMO 的连续时间系统或者离散时间系统。

impulse(sys,t)定义计算时的时间矢量。用户可以指定一个仿真终止时间，这时 t 为一个标量；也可以通过诸如 t = 0:dt:Tfinal 命令设置一个时间矢量。对于离散系统，时间间隔 dt 必须与采样周期匹配。

impulse(sys1,sys2,...,sysN)和 impulse(sys1,sys2,...,sysN,t)同时仿真多个 LTI 对象。

impulse(sys1,'PlotStyle1',...,sysN,'PlotStyleN')定义每个仿真绘制的绘制属性。其中 PlotStyle1 和 PlotStyleN 为 Matlab 标准命令 plot 支持的各种属性标识字符串。

[y,t,x] = impulse(sys)、 [y,t] = impulse(sys)和[y] = impulse(sys,t)计算仿真数据，且不在窗口上显示。其中，y 为输出响应矢量；t 为时间矢量；x 为状态轨迹

数据。对于 MIMO 系统，y 的维数为：(length of t)*(number of outputs)*(number of inputs)，y(:,:,j)对应于第 j 个输入通道；同样地，x 的维数为：(length of t)*(number of states)*(number of inputs)。

举例：绘制如下系统的脉冲响应，见图 2.7.2。

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}-0.5572 & -0.7814\\ 0.7814 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}1 & -1\\ 0 & 2\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix}$$

$$y=\begin{bmatrix}1.9691 & 6.4493\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

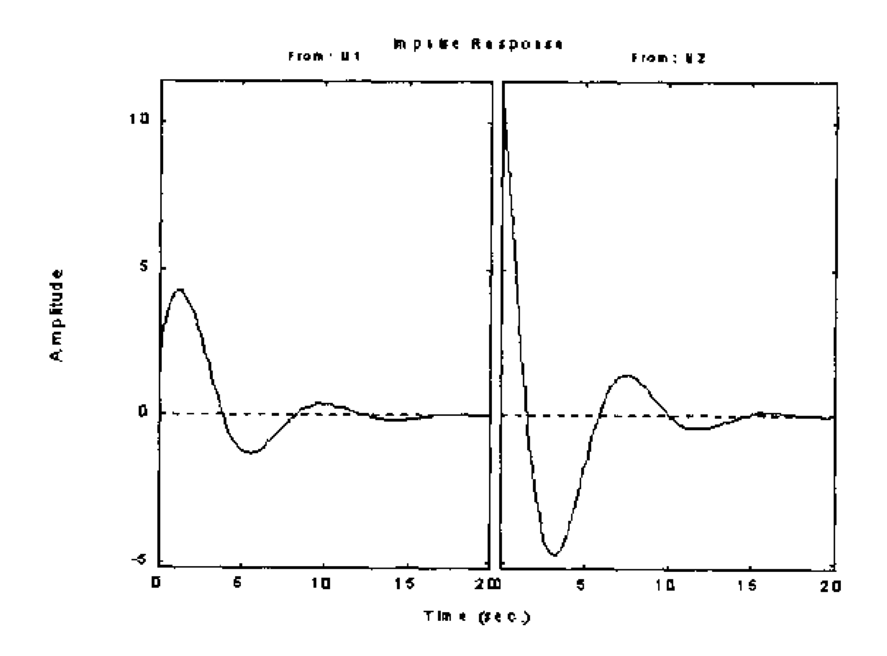


图 2.7.2 系统的脉冲响应

```
a = [-0.5572 -0.7814;0.7814 0];
b = [1 -1;0 2];
c = [1.9691 6.4493];
sys = ss(a,b,c,0);
impulse(sys)
```

3 initial

功能：计算系统零输入响应。

格式：initial(sys,x0)

initial(sys,x0,t)

initial(sys1,sys2,...,sysN,x0)

initial(sys1,sys2,...,sysN,x0,t)

initial(sys1,'PlotStyle1',...,sysN,'PlotStyleN',x0)

[y,t,x] = initial(sys,x0)

说明：initial 函数用于计算系统的零输入响应。当调用无输出变量时，initial 在当前图形窗口中直接绘出系统的单位冲激响应。

initial(sys,x0) 计算并在当前窗口绘制 LTI 对象 sys 的零输入响应，可应用于 SISO 或者 MIMO 的连续时间系统或者离散时间系统。

initial(sys,x0,t) 定义计算时的时间矢量。用户可以指定一个仿真终止时间，这时 t 为一个标量；也可以通过诸如 t = 0:dt:Tfinal 命令设置一个时间矢量。对于离散系统，时间间隔 dt 必须与采样周期匹配。

initial(sys1,sys2,...,sysN,x0)和 initial(sys1,sys2,...,sysN,x0,t) 同时仿真多个 LTI 对象。

initial(sys1,'PlotStyle1',...,sysN,'PlotStyleN',x0) 定义每个仿真绘制的绘制属性。其中 PlotStyle1 和 PlotStyleN 为 Matlab 标准命令 plot 支持的各种属性标识字符串。

[y,t,x] = initial(sys,x0)和[y,t,x] = initial(sys,x0,t) 计算仿真数据，且不在窗口上显示。其中，y 为输出响应矢量；t 为时间矢量；x 为状态轨迹数据。

举例：绘制如下系统的零输入响应，见图 2.7.3。

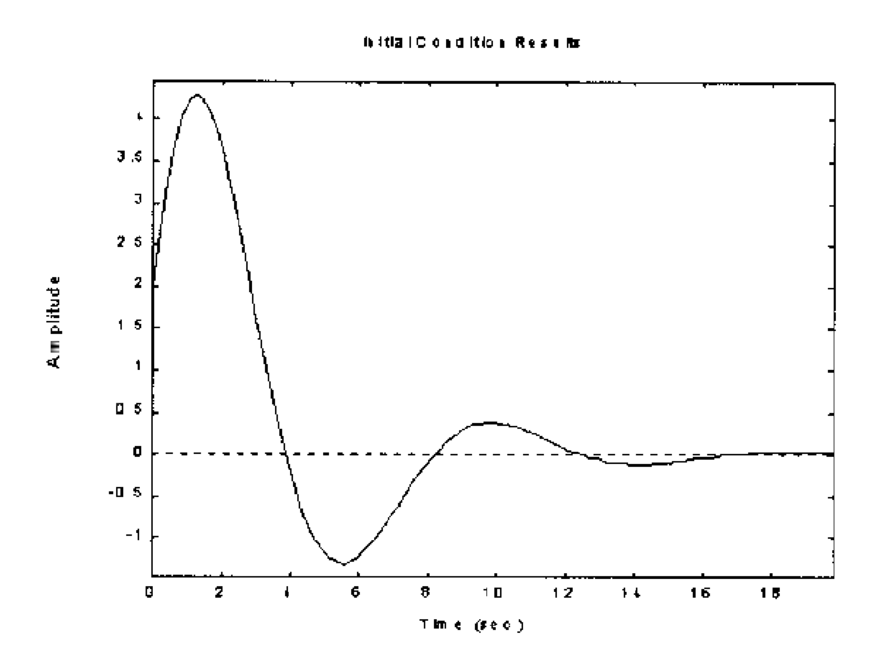


图 2.7.3　系统的零输入响应

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}-0.5572 & -0.7814\\ 0.7814 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

$$y=\begin{bmatrix}1.9691 & 6.4493\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

$$x0=\begin{bmatrix}1\\0\end{bmatrix}$$

```
a = [-0.5572 -0.7814;0.7814 0];
c = [1.9691 6.4493];
x0 = [1 ; 0]
sys = ss(a,[],c,[]);
initial(sys,x0)
```

4 lsim

功能： 对系统的任意输入进行仿真。

格式： lsim(sys,u,t)

lsim(sys,u,t,x0)

lsim(sys1,sys2,...,sysN,u,t)

lsim(sys1,sys2,...,sysN,u,t,x0)

lsim(sys1,'PlotStyle1',...,sysN,'PlotStyleN',u,t)

[y,t,x] = lsim(sys,u,t,x0)

说明： lsim 函数用于计算系统的任意输入响应。当调用无输出变量时，lsim 在当前图形窗口中直接绘出系统的单位冲激响应。

lsim(sys,u,t) 计算并在当前窗口绘制 LTI 对象 sys 在输入为 u(t)时的响应，可用于 SISO 或者 MIMO 的连续时间系统或者离散时间系统。

lsim(sys,u,t,x0)定义系统的初始状态 x0。

lsim(sys1,sys2,...,sysN,u,t)和 lsim(sys1,sys2,...,sysN,u,t,x0) 同时仿真多个 LTI 对象。

lsim(sys1,'PlotStyle1',...,sysN,'PlotStyleN',u,t) 定义每个仿真绘制的绘制属性。其中 PlotStyle1 和 PlotStyleN 为 Matlab 标准命令 plot 支持的各种属性标识字符串。

[y,t] = lsim(sys,u,t)、[y,t,x] = lsim(sys,u,t)和[y,t,x] = lsim(sys,u,t,x0)计算仿真数据，且不在窗口上显示。

举例： 计算下述系统的方波响应。其中，方波的周期为 4 秒，持续时间为 10 秒，采样周期为 0.1 秒，见图 2.7.4。

$$H(s)=\begin{bmatrix}\dfrac{2s^2+5s+1}{s^2+2s+3}\\[2mm]\dfrac{s-1}{s^2+s+5}\end{bmatrix}$$

```
[u,t] = gensig('square',4,10,0.1)
h= [tf([2 5 1],[1 2 3]) ; tf([1 -1],[1 1 5])]
lsim(h,u,t)
```



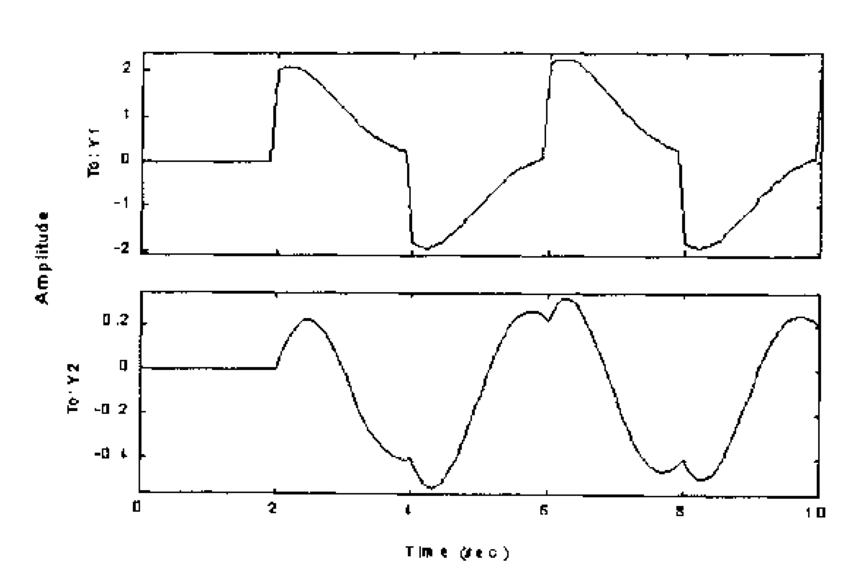


图 2.7.4　系统的方波响应

5　step

功能： 计算系统阶跃响应。

格式： step(sys)

step(sys,t)

step(sys1,sys2,...,sysN)

step(sys1,sys2,...,sysN,t)

step(sys1,'PlotStyle1',...,sysN,'PlotStyleN')

[y,t,x] = step(sys)

说明： step 函数用于计算系统的阶跃响应。当调用无输出变量时，step 在当前图形窗口中直接绘出系统的阶跃响应。

step(sys) 计算并在当前窗口绘制 LTI 对象 sys 的阶跃响应，可应用于 SISO 或者 MIMO 的连续时间系统或者离散时间系统。

step(sys,t) 定义计算时的时间矢量。用户可以指定一个仿真终止时间，这时 t 为一个标量；也可以通过诸如 t = 0:dt:Tfinal 命令设置一个时间矢量。对于离散系统，时间间隔 dt 必须与采样周期匹配。

step(sys1,sys2,...,sysN)和 step(sys1,sys2,...,sysN,t) 同时仿真多个 LTI 对象。

step(sys1,'PlotStyle1',...,sysN,'PlotStyleN') 定义每个仿真绘制的绘制属性。其中 PlotStyle1 和 PlotStyleN 为 Matlab 标准命令 plot 支持的各种属性标识字符串。

[y,t,x] = step(sys) 计算仿真数据，且不在窗口上显示。其中，y 为输出响应矢量；t 为时间矢量；x 为状态轨迹数据。对于 MIMO 系统，y 的维数为：(length of t)*(number of outputs)*(number of (inputs)，y(:,:,j)对应于第 j 个输入通道；同样地，x 的维数为：(length of t)*(number of states)*(number of　inputs)。

举例：绘制如下系统的脉冲响应,见图 2.7.5。

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}-0.5572 & -0.7814\\ 0.7814 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}1 & -1\\ 0 & 2\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix}$$

$$y=\begin{bmatrix}1.9691 & 6.4493\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

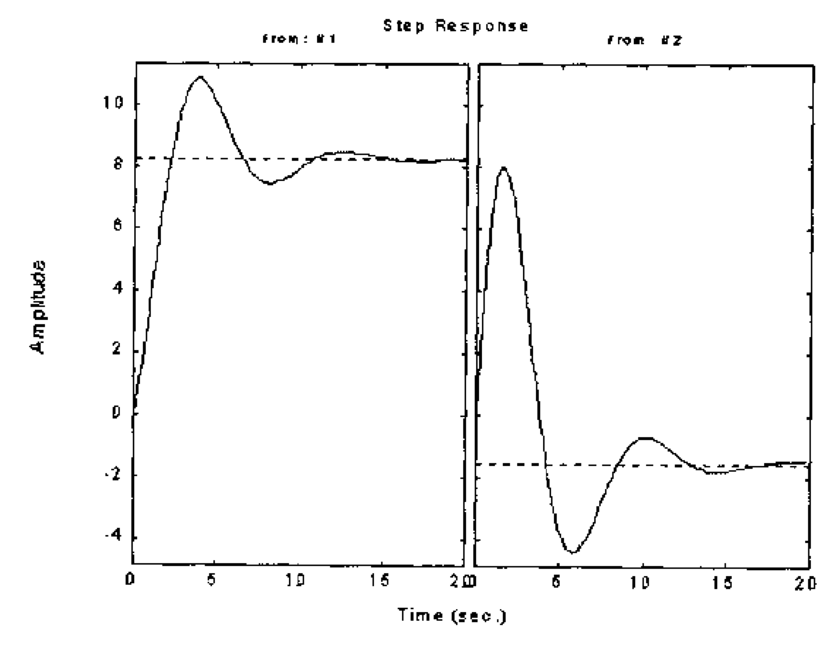


图 2.7.5 系统的阶跃响应

```
a = [-0.5572 -0.7814;0.7814 0];
b = [1 -1;0 2];
c = [1.9691 6.4493];
sys = ss(a,b,c,0);
step(sys)
```

2.8 极点配置与状态观测器设计

2.8.1 极点配置和状态观测

1 极点配置

- 极点配置问题描述

设 LTI 系统

$$\dot{x}=Ax+Bu$$

状态反馈极点配置即确定状态反馈控制

$u = -Kx + v$

使状态反馈闭环系统

$$\dot{x} = (A - BK)x + Bv$$

的极点为$\{\lambda_1^* \quad \lambda_2^* \quad \cdots \quad \lambda_n^*\}$。

可以证明，对给定系统可以进行任意极点配置的充要条件是系统为完全能控。

- 极点配置算法

（1）相似转换法

相似转换法通过求取一个相似转换矩阵 T，使系统具有期望的极点。

第 1 步　计算 A 的特征多项式

$$\det(SI - A) = s^n + a_{n-1}s^{n-1} + \cdots + a_1 s^1 + a_0$$

第 2 步　计算由$\{\lambda_1^* \quad \lambda_2^* \quad \cdots \quad \lambda_n^*\}$确定的多项式

$$\alpha^*(s) = (s - \lambda_1^*)(s - \lambda_2^*)\cdots(s - \lambda_n^*) = s^n + a_{n-1}^* s^{n-1} + \cdots + a_1^* s^1 + a_0^*$$

第 3 步　计算

$$K = [a_0^* - a_0, \quad a_1^* - a_1, \quad \cdots, \quad a_{n-1}^* - a_{n-1},]$$

第 4 步　计算变换阵 P

$$P = [A^{n-1}B, \cdots, AB, B]\begin{bmatrix} 1 & & & 0 \\ a_{n-1} & \ddots & & \\ \vdots & & \ddots & \\ a_1 & \cdots & a_{n-1} & 1 \end{bmatrix}$$

第 5 步　求 $T=P^{-1}$

第 6 步　求反馈增益矩阵 K

$K = \overline{K}T$

（2）Ackermann 公式，该方法通过计算矩阵 A 的特征多项式来求取反馈增益矩阵 K。步骤如下：

第 1 步　确定矩阵特征多项式的系数

$$\alpha^*(s) = (s - \lambda_1^*)(s - \lambda_2^*)\cdots(s - \lambda_n^*) = s^n + a_{n-1}^* s^{n-1} + \cdots + a_1^* s^1 + a_0^*$$

第 2 步　求

$$\phi(A) = A^n + a_{n-1}^* A^{n-1} + \cdots + a_1^* A^1 + a_0^* I$$

第 3 步　计算增益矩阵 K

$$K=(0\,0\cdots 0\,1)(B,AB,\cdots A^{n-1}B)\phi(A)$$

2 状态观测器

- 全维状态观测器的问题提出

对于 LTI 系统

$$\dot{x}=Ax+Bu$$

$$y=Cx$$

全维状态观测器就是以输出 y 和输入 u 为输入，输出 $\hat{x}$ 满足

$$\lim_{t\to\infty}\hat{x}(t)=\lim_{t\to\infty}x(t)$$

通常，全维观测器采用下述模型，见图 2.8.1。此时，全维观测器的动态方程为

$$\dot{\hat{x}}=A\hat{x}+Bu+L(y-C\hat{x})=(A-LC)\hat{x}+Bu+Ly$$

可以证明，对于上述 LTI 系统，若系统完全能观，则必然可由上式表述的全维观测器来重构其状态，并可以通过选择增益矩阵 L 而任意配置观测器的全部特征值。

- 全维观测器的设计算法

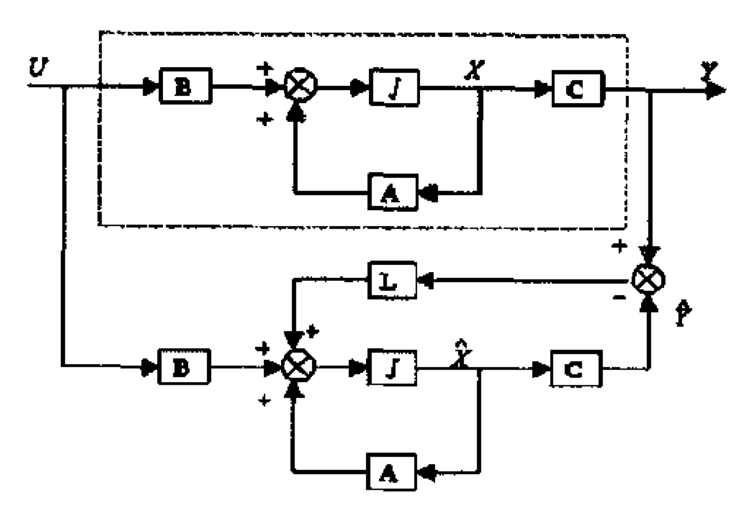


图 2.8.1 全维观测器

设系统完全能观，全维观测器的期望极点为$\{\lambda_1^* \quad \lambda_2^* \quad \cdots \quad \lambda_n^*\}$，则计算步骤为：

第 1 步 导出对偶系统$\{A^T,\ C^T,\ B^T\}$

第 2 步 利用极点配置算法，对对偶系统确定反馈增益矩阵 K，使系统的极点为

$$\{\lambda_1^* \quad \lambda_2^* \quad \cdots \quad \lambda_n^*\}$$

第 3 步 取 $L=K^T$。

第 4 步 计算 $A-LC$，得到全维观测器

$$\dot{\hat{x}}=(A-LC)\hat{x}+Bu+Ly$$

3 降维观测器

考虑到系统的输出中已经包含系统状态的部分信息，因此在直接利用这部分信息的基础上，可以构造出低于被估系统的状态观测器，这就是降维观测器。

可以证明，对于 LTI 系统，设系统为完全能观，且矩阵 C 的秩 rank(C)=q，则可以构造(n-q)维的状态观测器。关于降维观测器设计的算法，请详见参考文献[2]。

2.8.2 极点配置和状态观测函数

关于极点配置和状态观测的函数，见表 2.8.1。

表 2.8.1 极点配置与状态观测器设计函数列表

函数名称	功能
acker	SISO 极点配置
place	MIMO 极点配置
estim	生成系统状态估计器或观测器
reg	系统调节器(regulator)生成

1 acker

功能：SISO 极点配置。

格式：k = acker(a,b,p)

说明：acker 函数利用 Ackermann 公式计算反馈增益矩阵 K，使采用全反馈 $u=-Kx$ 的单输入系统具有指定的闭环极点 p，即

$$p = eig(A-BK)$$

k = acker(a,b,p) SISO 系统极点配置。其中 a,b,p,k 如上所示。

acker 函数也可以用来计算估计器增益(estimator gain)，这时应采用 l = acker(a',c',p).'调用。

2 place

功能：MIMO 极点配置。

格式：K = place(A,B,p)

[K,prec,message] = place(A,B,p)

说明：place 函数利用 Ackermann 公式计算反馈增益矩阵 K，使采用全反馈 $u=-Kx$ 的多输入系统具有指定的闭环极点 p，即

$$p = eig(A-BK)$$

K = place(A,B,p) MIMO 极点配置。其中 A,B,p,K 如上所述。

[K,prec,message] = place(A,B,p)同时返回系统闭环实际极点与希望极点 p 的接近程度 prec。prec 中的每个量的值为匹配的位数。如果系统闭环的实际极点偏离希望极点 10% 以上，则 message 将给出警告信息。

place 函数也可以用来计算估计器增益(estimator gain)，这时应采用 l = place(A',C',p).'调用。

举例：考虑一个 2 输入、3 输出、3 状态的系统(a,b,c,d)，今将系统的极点配置至 p = [1.1 23 5.0]

```
p = [1 1.23 5.0];
K = place(a,b,p)
```

3 estim

功能：生成系统状态估计器(estimator)。

格式：est = estim(sys,L)

est = estim(sys,L,sensors,known)

说明：estim 从 LTI 系统的状态空间模型和增益矩阵 L 中生成系统的状态估计器。增益矩阵 L 可由极点配置函数 place 形成，或者由 Kalman 滤波函数 Kalman 生成。

est = estim(sys,L)生成给定增益矩阵 L 下的状态空间模型的 LTI 系统 sys 的状态和输出估计器 est，并假定 sys 的所有输入为随机的(系统或者测量噪声)，系统的所有输出可测。即对于下述连续时间系统

$$\dot{x} = Ax + Bw$$

$$y = Cx + Dw$$

estim 函数将生成下述状态和输出估计器：

$$\dot{\hat{x}} = A\hat{x} + L(y - C\hat{x})$$

$$\begin{bmatrix} \hat{y} \\ \hat{x} \end{bmatrix} = \begin{bmatrix} C \\ I \end{bmatrix} \hat{x}$$

对于离散时间系统，状态和输出估计器具有类似的函数。

est = estim(sys,L,sensors,known)生成更为普遍的 LTI 系统 sys 的状态和输出估计器。其中，sensors 用于指定可测输出，known 用于指定已知输入。这里假定系统同时具有已知输入 u 和系统噪声 w，系统输出为非完全可测。即对于下述连续时间系统

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} x + \begin{bmatrix} D_{11} \\ D_{21} \end{bmatrix} w + \begin{bmatrix} D_{12} \\ D_{22} \end{bmatrix} u$$

$$\dot{x} = Ax + B_1 w + B_2 u$$

estim 函数将生成下述状态和输出估计器：

$$\dot{\hat{x}} = A\hat{x} + B_2 u + L(y - C_2\hat{x} - D_{22}u)$$

$$\begin{bmatrix} \hat{y} \\ \hat{x} \end{bmatrix} = \begin{bmatrix} C_2 \\ I \end{bmatrix} \hat{x} + \begin{bmatrix} D_{22} \\ 0 \end{bmatrix} u$$

举例：考虑一个由 7 输出和 4 输入构成的状态空间系统(a, b, c, d)，当系统的输出 4、7 和 1 作为传感器输出，输入 1、4 和 3 作为已知输入时，设计卡尔曼增益矩

阵 L，从而得到状态估计器，这时 Matlab 程序为

```
sensors = [4,7,1];
known = [1,4,3];
est = estim(sys,L,sensors,known)
```

4　reg

功能：系统调节器(regulator)生成。

格式：rsys = reg(sys,K,L)

rsys = reg(sys,K,L,sensors,known,controls)

说明：rsys = reg(sys,K,L) 生成给定状态估计增益矩阵 L 及状态反馈增益矩阵 K 下的状态空间模型的 LTI 系统 sys 的调节器或者补偿器 est，并假定系统的所有输出可测。即对于下述连续时间系统

$$\dot{x} = Ax + Bw$$

$$y = Cx + Dw$$

reg 函数将生成下述调节器，见图 2.8.2。

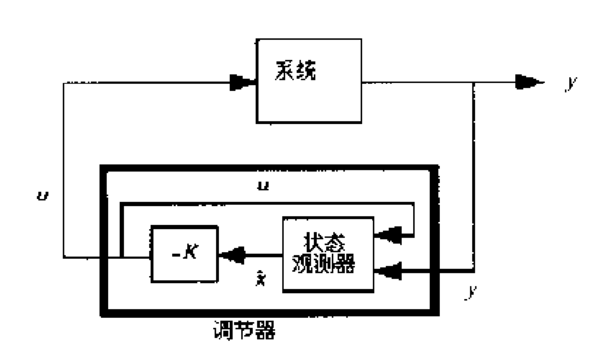


图 2.8.2　调节器

$$\dot{\hat{x}} = [A - LC - (B - LD)K]\hat{x} + Ly$$

$$u = -K\hat{x}$$

rsys = reg(sys,K,L,sensors,known,controls) 生成更为普遍的 LTI 系统 sys 的状态和输出估计器。其中，sensors 用于指定可测输出，known 用于指定已知输入。controls 用于指定控制输入。这里假定系统同时具有已知输入 u_d，控制输入 u 和系统噪声 w，系统输出为非完全可测。生成的调节器结构如图 2.8.3 所示。

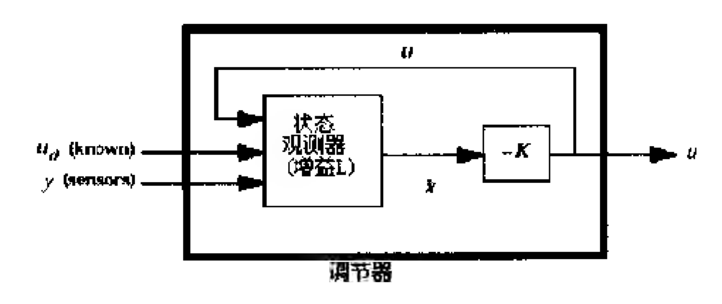


图 2.8.3　扩展的调节器结构

举例：考虑一个由 7 输出/4 输入的连续系统(a, b, c, d)，利用受控对象的第 1、2、4 个输入作为控制输入构成增益矩阵 k，利用第 4、7、1 个输出作为传感器输出构成 kalman 增益矩阵 l，输入 3 构成已知输入。这时 Matlab 程序为

```
controls = [1,2,4];
sensors = [4,7,1];
known = [3];
regulator = reg(sys,K,L,sensors,known,controls)
```

2.9 LQ 最优控制

2.9.1 LQ 最优控制问题

1 问题提出

设 LTI 系统

$$\dot{x} = Ax + Bu \quad x(0) = x_0 \quad t \in [0, \quad t_f]$$

二次型性能指标函数

$$J(u(\cdot)) = \frac{1}{2}x^T(t_f)Sx(t_f) + \frac{1}{2}\int_0^{t_f} x^T(t)Qx(t) + u^T(t)Qu(t)dt$$

其中，$x(t_f)$为末状态，矩阵 S，Q 为半正定对称矩阵，R 为正定对称矩阵。

所谓 LQ 最优控制问题，就是寻找一个控制 u^*，使系统沿初态 $x0$ 出发的相应轨迹 x^*，其性能指标满足

$$J(u^*(\cdot)) = \min_{u()} J(u(\cdot))$$

则称 u^*为 LQ 问题的最优控制，x^*为相应的最优控制，$J(u^*)$为最优性能指标。

可以将 LQ 问题分为有限时间和无限时间 LQ 问题。有限时间 LQ 问题中，末时刻 t_f固定且为有限值；而在无限时间 LQ 问题中，末时刻 $t_f=\infty$。

工程上还将 LQ 最优控制问题分为调节问题和跟踪问题。调节问题就是求取最优控制，使在其作用下将系统由初始状态驱动到零平衡状态，同时使性能指标函数最小；跟踪问题则要求系统输出跟踪已知或者未知的参考信号的同时，使性能指标最小。事实上，跟踪问题是调节问题的推广。

2 有限时间LQ调节问题

对于上述连续时间 LTI 系统的有限时间 LQ 调节问题，可以证明，u^*为其最优控制的充要条件是

$$u^*(t)=-K^*x^*(t)\quad K^*(t)=-R^{-1}B^TP(t)$$

最优轨迹 x^* 为下述状态方程的解

$$\dot{x}^*(t)=Ax^*(t)+Bu^*(t)\quad x^*(0)=x_0$$

最优性能指标为

$$J^*=\frac{1}{2}x_0^TP(0)x_0$$

其中，$P(t)$ 为下述 Riccati 微分方程的正半定对称解矩阵。

$$-\dot{P}(t)=A^TP(t)+P(t)A-P(t)BR^{-1}B^TP(t)+Q\quad P(t_f)=s$$

其最优调节系统是一个状态反馈系统，反馈阵唯一，最优调节系统的动态方程为：

$$\dot{x}^*=(A-BR^{-1}B^TP(t))x^*\quad x^*(0)=x_0\quad t\in[0,\ t_f]$$

3　无限时间LQ调节问题

对于无限时间 LQ 调节问题，由于调节问题中平衡状态为零状态且一个最优控制系统应当是稳定的，故此时性能指标中常常不考虑相对于末状态的二次项，于是问题可描述为：

设 LTI 系统

$$\dot{x}=Ax+Bu$$

二次型性能指标函数

$$J(u(\cdot))=\int_0^\infty(x^T(t)Qx(t)+u^T(t)Ru(t))dt$$

可以证明，对于上述问题，u^* 为其最优控制的充要条件是

$$u^*(t)=-K^*x^*\quad K^*=-R^{-1}B^TP$$

最优轨迹 x^* 为下述状态方程的解：

$$\dot{x}^*(t)=Ax^*(t)+Bu^*(t)\quad x^*(0)=x_0$$

最优性能指标为

$$J^*=x_0^TPx_0$$

其中，$P(t)$ 为下述 Riccati 代数方程的正定对称解矩阵。

$$A^TP+PA-PBR^{-1}B^TP+Q=0$$

其最犹调节系统是一个状态反馈系统，且保持定常，反馈阵唯一，最优调节系统的动态方程为

$$\dot{x}^*=(A-BR^{-1}B^TP)x^*\quad x^*(0)=x_0$$

2.9.2　LQG 最优控制问题

1　线性二次型Gauss最优控制问题描述

设 LTI 系统中含有系统噪声和测量噪声，即系统为

$$\dot{x}(t)=Ax(t)+Bu(t)+Tw(t)$$
$$y(t)=Cx(t)+v(t)$$

其中，$w(t)$，$v(t)$均为零均值的 Gauss 白噪声，且有

$$E(w)=E(v)=0 \quad E\left(ww^T\right)=Q \quad E\left(vv^T\right)=R \quad W\left(wv^T\right)=0$$

则 LQG 最优控制问题就是设计控制输入 $u(t)$，使二次性能指标函数

$$J(u(\cdot))=E\left(\int_0^\infty \left(x^T(t)Qx(t)+u^T(t)Ru(t)\right)dt\right)$$

最小。

2　LQG 问题的求解

根据分离原理，LQG 问题可以分解为 2 个子问题求解。

(1) 利用 Kalman 滤波理论，从状态 $x(t)$中得到最优估计，即使

$$P=\lim_{t\to\infty}E\left(\{x-\hat{x}\}\{x-\hat{x}\}^T\right)$$

最小。

可以证明，这种最优估计器的动态方程为

$$\dot{\hat{x}}=A\hat{x}+Bu+K_1\left(y-C\hat{x}\right)$$

其中，滤波器增益 K_1 为：

$$K_1=PC^TV^{-1}$$

P 满足下述代数 Raccati 方程

$$A^TP+PA-PC^TR^{-1}CP+TWT^T=0$$

(2) 利用估计状态 $\hat{x}$ 代替 x，设计满足二次性能指标的 LQ 控制律。根据上节关于 LQ 问题的讨论，有

$$u^*(t)=-K^*x^*(t) \quad K^*=-R^{-1}B^TP$$

其中，P 满足下述代数 Raccati 方程

$$A^TP+PA-PBR^{-1}B^TP+Q=0$$

2.9.3 最优控制函数

关于 LQ 和 LQG 最优控制的函数，见表 2.9.1。

表 2.9.1 LQ 和 LQG 最优控制函数列表

函数名称	功能
lqr	连续系统的 LQ 调节器设计
dlqr	离散系统的 LQ 调节器设计
lqry	系统的 LQ 调节器设计
lqrd	计算连续时间系统的离散 LQ 调节器设计
kalman	系统的 Kalman 滤波器设计
kalmd	连续系统的离散 Kalman 滤波器设计
lqgreg	根据 kalman 估计器增益和状态反馈增益建立 LQG 调节器

1 lqr

功能：连续系统的 LQ 调节器设计。

格式：[K,S,e] = lqr(A,B,Q,R)

[K,S,e] = lqr(A,B,Q,R,N)

说明：[K,S,e] = lqr(A,B,Q,R,N)计算连续时间系统的最优反馈增益矩阵 K，使系统

$$\dot{x} = Ax + Bu$$

采用反馈律

$$u = -Kx$$

使性能指标函数

$$J = \int_0^\infty \left(x^T Qx + u^T Ru + 2x^T Nu\right)dt$$

最小。同时返回代数 Raccati 方程

$$A^T S + SA - (SB + N)R^{-1}\left(B^T S + N^T\right) + Q = 0$$

的解 S 及闭环系统的特征值 e。缺省时 N=0。这里

$$K = R^{-1}\left(B^T S + N^T\right)$$

2 dlqr

功能：离散系统的 LQ 调节器设计。

格式：[K,S,e] = dlqr(a,b,Q,R)

[K,S,e] = dlqr(a,b,Q,R,N)

说明：[K,S,e] = dlqr(a,b,Q,R,N) 计算离散时间系统的最优反馈增益矩阵 K，使系统

$$x[n+1] = Ax[n] + Bu[n]$$

采用反馈律

$$u[n] = -Kx[n]$$

使性能指标函数

$$J = \sum_{n=1}^{\infty} \left(x[n]^T Qx[n] + u[n]^T Ru[n] + 2x[n]^T Nu[n] \right)$$

最小。同时返回代数 Raccati 方程

$$A^T SA + A - \left(A^T SB + N\right)\left(B^T XB + R\right)^{-1}\left(B^T SA + N^T\right) + Q = 0$$

的解 S 及闭环系统的特征值 e。缺省时 N=0。这里

$$K = \left(B^T XB + R\right)^{-1}\left(B^T SA + N^T\right)$$

3 lqry

功能：系统的 LQ 调节器设计。

格式：[K,S,e] = lqry(sys,Q,R)

[K,S,e] = lqry(sys,Q,R,N)

说明：[K,S,e] = lqry(sys,Q,R,N)设计系统的最优反馈增益矩阵 K，使系统

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

采用反馈律

$$u = -Kx$$

使性能指标函数

$$J = \int_0^{\infty} \left(y^T Qy + u^T Ru + 2y^T Nu \right) dt$$

最小。同时返回代数 Raccati 方程的解 S 及闭环系统的特征值 e。缺省时 N=0。该函数也可以计算相应的离散时间系统的最优反馈增益矩阵 K。

4 lqrd

功能：连续时间系统的离散 LQ 调节器设计。

格式：[Kd,S,e] = lqrd(A,B,Q,R,Ts)

[Kd,S,e] = lqrd(A,B,Q,R,N,Ts)

说明：[Kd,S,e] = lqrd(A,B,Q,R,N,Ts) 设计连续时间系统的离散最优反馈增益矩阵 K，使系统

$$x[n+1] = Ax[n] + Bu[n]$$

采用反馈律

$$u[n] = -K_d x[n]$$

使性能指标函数

$$J=\int_0^{\infty}\left(x^TQx+u^TRu+2x^TNu\right)dt$$

最小。同时返回代数 Raccati 方程的解 S 及闭环系统的特征值 e。缺省时 N=0。

5 kalman

功能： 系统的 Kalman 滤波器设计。

格式： [kest,L,P] = kalman(sys,Qn,Rn,Nn)

[kest,L,P,M,Z] = kalman(sys,Qn,Rn,Nn)　%仅用于离散时间系统

[kest,L,P] = kalman(sys,Qn,Rn,Nn,sensors,known)

说明： kalman 函数用于对带有系统噪声和测量噪声的连续时间或者离散时间系统设计一个 Kalman 最优状态估计器。

对于下述连续时间系统

$$\dot{x}=Ax+Bu+Gw$$
$$y_v=Cx+Du+Hw+v$$

其中，w,v 满足

$$E(w)=E(v)=0\quad E\left(ww^T\right)=Q\quad E\left(vv^T\right)=R\quad W\left(wv^T\right)=N$$

则由 Kalman 滤波器构成的状态估计器

$$\dot{\hat{x}}=A\hat{x}+Bu+L\left(y-C\hat{x}-Du\right)$$

$$\begin{bmatrix}\hat{y}\\ \hat{x}\end{bmatrix}=\begin{bmatrix}C\\ I\end{bmatrix}\hat{x}+\begin{bmatrix}D\\ 0\end{bmatrix}u$$

在最小化条件

$$P=\lim_{t\to\infty}E\left(\{x-\hat{x}\}\{x-\hat{x}\}^T\right)$$

下，为最优状态估计器。

这里增益矩阵 L 为下述代数 Raccati 方程的解。

Kalman 滤波器结构见图 2.9.1。

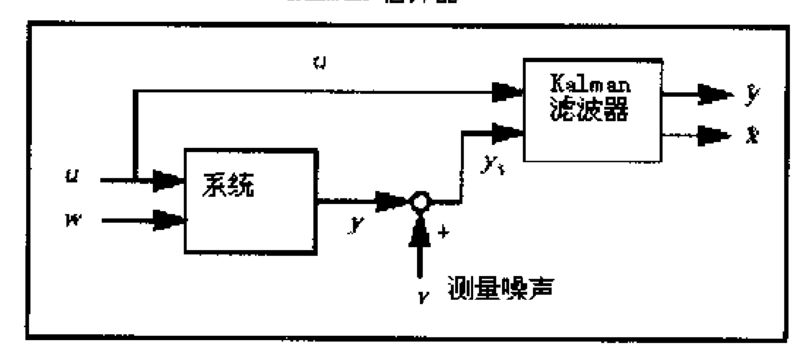


图 2.9.1　Kalman 滤波器结构

对于离散时间系统：

$$x[n+1]=Ax[n]+Bu[n]+Gw[n]$$
$$y_v[n]=Cx[n]+Du[n]+Hw[n]+v[n]$$

其中，w,v 满足

$$E(w[n])=E(v[n])=0 \quad E\left(w[n]w[n]^T\right)=Q$$
$$E\left(v[n]v[n]^T\right)=R \quad W\left(w[n]v[n]^T\right)=N$$

则由 Kalman 滤波器构成的状态估计器

$$\hat{x}[n+1n]=A\hat{x}[n|n-1]+Bu[n]+L(y_v[n]-C\hat{x}[n|n-1]-Du[n])$$
$$\begin{bmatrix}\hat{y}[n|n]\\ \hat{x}[n|n]\end{bmatrix}=\begin{bmatrix}C(I-MC)\\ I-MC\end{bmatrix}\hat{x}[n|n-1]+\begin{bmatrix}(I-MC)D & CM\\ -MD & M\end{bmatrix}\begin{bmatrix}u[n]\\ y_v[n]\end{bmatrix}$$

在最小化条件

$$P=\lim_{n\to\infty}E\left(e[n|n-1]e[n|n-1]^T\right) \quad e[n|n-1]=x[n]-x[n|n-1]$$

下，为最优状态估计器。

这里增益矩阵 M 为下述代数 Raccati 方程的解。

[kest,L,P] = kalman(sys,Qn,Rn,Nn)返回上述系统的 Kalman 状态估计器 kest。其中 sys 为原系统的状态空间模型。Qn、Rn 和 Nn 上述的 Q、R 和 N 矩阵。返回值中 kest 为系统的状态估计器，L 和 P 对应于上述的和 P 矩阵。

[kest,L,P,M,Z] = kalman(sys,Qn,Rn,Nn) 仅适应于离散时间系统的滤波器设计，同时返回矩阵 Z。Z 的定义为：

$$Z=\lim_{n\to\infty}E\left(e[n|n]e[n|n]^T\right) \quad e[n|n]=x[n]-\hat{x}[n|n]$$

[kest,L,P] = kalman(sys,Qn,Rn,Nn,sensors,known)生成更为普遍的 LTI 系统 sys 的 Kalman 状态估计器。其中，sensors 用于指定可测输出，known 用于指定已知输入，其余输入为噪声输入。这里假定系统已知输入 u 和噪声输入 w 混在一起，系统输出为非完全可测。

6 kalmd

功能： 连续系统的离散 Kalman 滤波器设计。

格式： [kest,L,P,M,Z] = kalmd(sys,Qn,Rn,Ts)

说明： [kest,L,P,M,Z] = kalmd(sys,Qn,Rn,Ts)用于对带有系统噪声和测量噪声的连续时间系统设计一个离散的 Kalman 最优状态估计器。

对于连续时间系统

$$\dot{x}=Ax+Bu+Gw$$
$$y_v=Cx+Du+Hw+v$$

其中，w,v 满足

$$E(w)=E(v)=0 \quad E(ww^T)=Q \quad E(vv^T)=R \quad W(wv^T)=N$$

kalmd 函数通过采用采样周期为 Ts 的零阶保持器离散化原连续时间系统，并通过将噪声协方差矩阵通过下式进行离散变换，将连续时间 Kalman 滤波器转换成等价的离散时间 Kalman 滤波器设计问题。然后计算该离散时间 Kalman 滤波器

$$Q_d=\int_0^{T_s} e^{At}GQG^T e^{A^T t}dt$$

$$R_d=R/T_s$$

其中，sys 为原连续时间系统的状态空间模型。Qn 和 Rn 分别对应于上述的 Q 和 R 矩阵。Ts 为离散化时的采样周期。返回值中 kest，L，P，M 和 Z 与 kalman 函数中的意义相同。

7 lqgreg

功能：根据 kalman 估计器增益和状态反馈增益建立 LQG 调节器。

格式：rlqg = lqgreg(kest,k)

rlqg = lqgreg(kest,k,'current')　%仅用于离散时间系统

rlqg = lqgreg(kest,k,controls)

说明：lqgreg 函数根据 kalman 估计器增益和状态反馈增益来建立 LQG 调节器。kalman 估计器增益常由函数 kalman 生成，状态反馈增益常由 lqr、dlqr 及 lqry 生成，见图 2.9.2。对于连续时间系统，其调节器方程为

$$u=-K\hat{x}$$

$$\dot{\hat{x}}=[A-LC-(B-LD)K]\hat{x}+Ly_v$$

对于离散时间系统，其调节器方程可以基于状态估计或者一步预测。

$$u[n]=-K\hat{x}[n|n-1]$$

$$u[n]=-K\hat{x}[n|n]$$

其中，$\hat{x}$ 为 Kalman 滤波器的状态估计，y_v 为系统的测量输出，详见 kalman 函数。

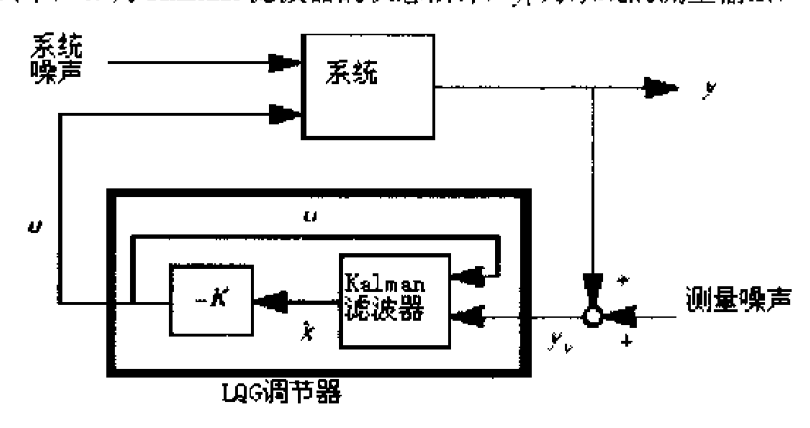


图 2.9.2　LQG 调节器

rlqg = lqgreg(kest,k)建立 LQG 调节器 rlqg。其中，kest 为 kalman 估计器增益，k 状态反馈增益。

rlqg = lqgreg(kest,k,'current') 仅用于离散时间系统，定义调节器使用状态估计。'current'使用状态估计计算。缺省时则采用一步预测。

rlqg = lqgreg(kest,k,controls)定义其他系统输入，见图 2.9.3 中的 u_d。

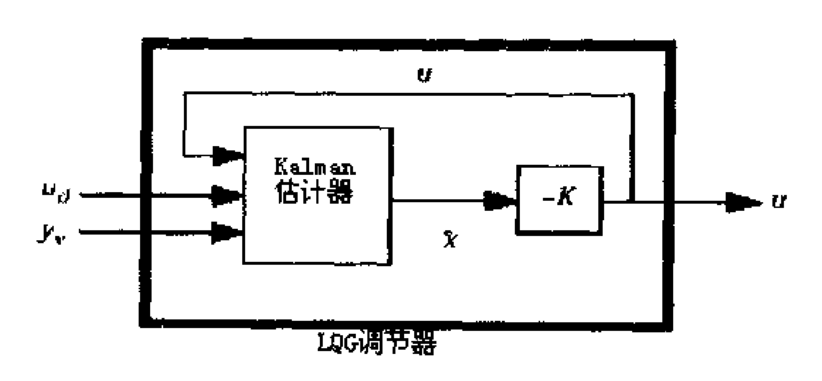


图 2.9.3　扩展 LQG 调节器

2.10　系统分析的 GUI 函数

除了提供大量的系统分析函数以外，新版本的控制系统工具箱还提供两个图形窗口分析工具 ltiview 和 rltool。

其中，ltiview 可用于分析 SISO 或者 MIMO 的 LTI 系统的各种响应，称作 LTI 观测器(LTI Viewer)；rltool 则用于系统的根轨迹设计。下面对其分别进行简单的介绍。关于其详细用法见联机文档。

2.10.1　ltiview 工具

LTI 观测器通过 `ltiview` 函数来激活。

1 ltiview

功能：激活 LTI 系统响应分析工具——LTI 观测器。

格式：

```
ltiview
ltiview(plottype,sys)
ltiview(plottype,sys,extras)
ltiview(plottype,sys1,sys2,...sysN)
ltiview(plottype,sys1,sys2,...sysN,extras)
ltiview(plottype,sys1,PlotStyle1,sys2,PlotStyle2,...)
```

说明：ltiview 激活一个新的 LTI 观测器。

ltiview(plottype,sys)定义想要分析的系统 sys 以及想要分析的系统响应类型 plottype。其中 plottype 可为以下字符串之一：

- 'step'——系统阶跃响应
- 'impulse'——系统脉冲响应
- 'initial'——系统零输入响应
- 'lsim'——系统的任意信号输入的时间响应
- 'bode'——系统 Bode 图
- 'nyquist'——系统 Nyquist 图
- 'nichols'——系统 Nichols 图
- 'sigma'——系统奇异值响应

ltiview(plottype,sys,extras)定义输入的附加参数。要注意附加参数根据系统的响应类型而不同。例如：ltiview('step',sys,Tfinal)定义了响应的终止时间 Tfinal。

ltiview(plottype,sys1,sys2,...sysN) 和 ltiview(plottype,sys1,sys2,...sysN,extras) 和 ltiview(plottype,sys1,PlotStyle1,sys2,PlotStyle2,...)允许 LTI 观测器同时分析多个系统的响应。

一个典型的 LTI 观测器窗口，如图 2.10.1 所示。可以将该窗口分为以下几个区。

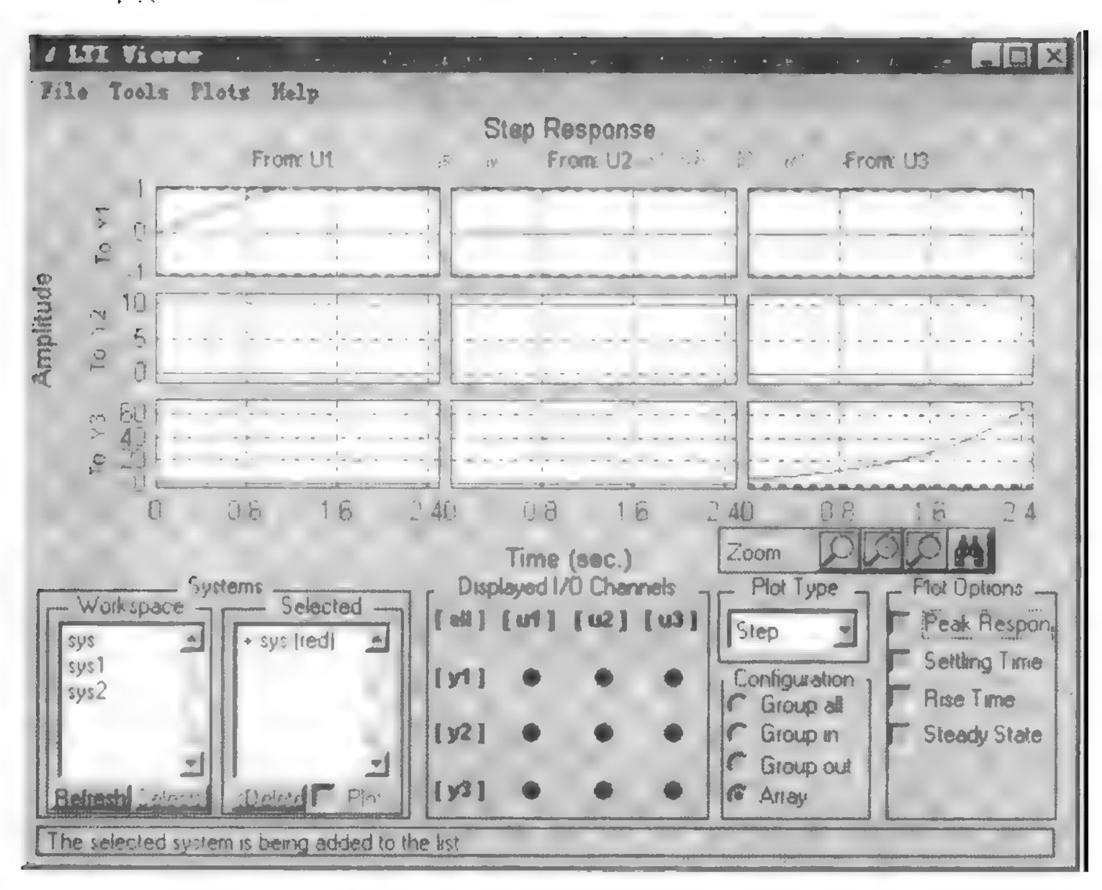


图 2.10.1　LTI 观测器窗口

其中：

- 菜单条允许用户通过选择不同的菜单项来获得一些附加的控制。
- 系统响应绘制区向用户展示了系统的响应曲线或者各种频域图绘制。用户可在输入输出通道显示选择区(Displayed I/O Channels)选择想要显示的I/O通道。在该例中显示了系统的所有通道的阶跃响应。
- 系统区(System)中的工作空间区(Workspace)用于从 Matlab 的工作空间中引入已经建立的系统。当工作空间中的系统有变化时，单击 Refresh 按钮可对 LTI 观测器进行更新。选择分析的系统后，单击 Select 按钮将把选择了的系统加进系统区(System)中的 Selected 区，并对该系统进行响应绘制。
- 输入输出通道显示选择区(Displayed I/O Channels)用于设置显示的I/O通道。可以以列(对应于输入)或者行(对应于输出)或者全选(选择 All)，也可以对单个 I/O 通道进行选择。对已经显示的 I/O 通道，其对应的圆点为红色。在该例中显示系统的所有通道的阶跃响应。
- 绘制类型区(Plot Type)用于选择系统的响应类型。下拉列表框中给出可选择的类型。下面的单选按钮允许用户进行更多的设置。
- 绘制选项区(Plot Options)中的复选框允许用户对绘制进行多个选择。
- 缩放工具栏(Zoom)给出了几个可对绘制进行缩放的按钮。

2.10.2 rltool 工具

根轨迹设计工具通过 `rltool` 函数来激活。

- rltool

功能：激活系统的根轨迹设计工具。

格式：rltool

rltool(sys)

rltool(sys,comp)

rltool(sys,comp,LocationFlag,FeedbackSign)

说明：rltool 激活一个新的根轨迹设计窗口，以允许用户对 SISO 系统进行补偿器设计。

rltool(sys) 激活一个新的根轨迹设计窗口。并且闭环系统的 P 模块为 sys，sys 可为任意存在于工作空间中的 LTI 系统。关于 P 模块的意义，见后面根轨迹设计窗口的说明。

rltool(sys,comp) 激活一个新的根轨迹设计窗口，同时定义补偿器的初值 comp。Comp 可为任意存在于工作空间中的 LTI 系统。

rltool(sys,comp,LocationFlag,FeedbackSign)同时定义补偿器的位置 LocationFlag 和反馈的类型 FeedbackSign。其中，当 LocationFlag=1 时，补偿器将位于前向通道中；如果 LocationFlag=2，则补偿器将位于反馈通道中。如果 FeedbackSign=-1，则反馈为负反馈；FeedbackSign=1 则为正反馈。缺省时

LocationFlag=1，FeedbackSign=-1。

一个典型的根轨迹设计窗口如图 2.10.2 所示。其中

- 菜单条允许用户引入或者引出系统模型，并可对其进行编辑。还可以对系统进行连续化或者离散化处理。
- 反馈结构图解区给出当前反馈系统的整个结构。K 为补偿器模块，P、H 和 F 为系统的其他模块，用户可以根据需要使用这几个模块构造实际系统。单击各模块可以观察模块的当前属性。左下角的按钮允许用户在正负反馈间进行切换。
- 补偿器描述区给出当前补偿器的结构描述。
- 根轨迹工具条上的按钮允许用户增加或者删除补偿器的零极点，并可对闭环系统的极点进行拖拽(Drag)。
- 增益编辑框允许用户改变增益值，从而改变闭环系统的极点位置。
- 绘制区用于显示系统的根轨迹。
- 坐标轴设置区(Axes setting)的按钮允许用户存取根轨迹绘制中的坐标轴设置。
- 缩放区(Zoom)按钮允许用户对根轨迹绘制进行缩放显示。
- 选中复选框区的任意一个将激活一个系统相应响应的分析窗口。

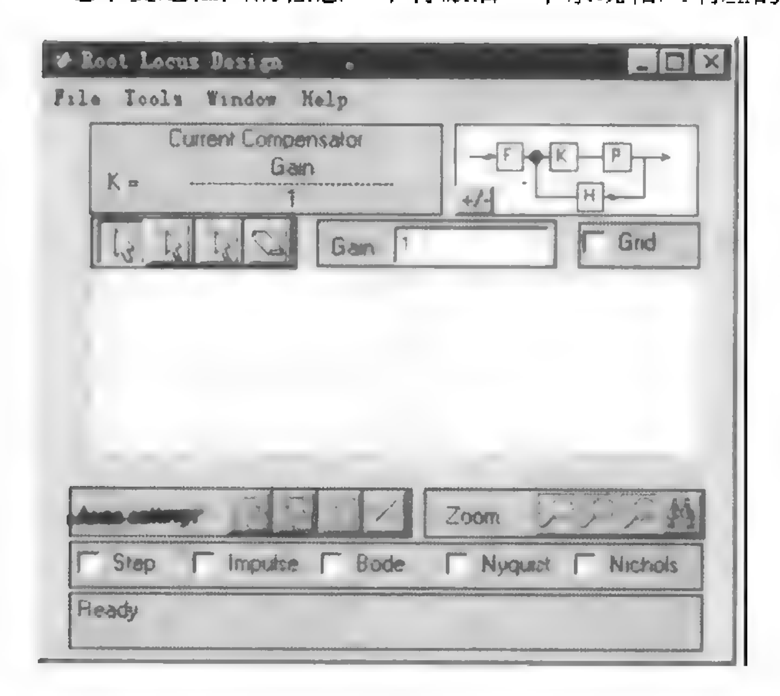


图 2.10.2　根轨迹设计窗口

参考文献

[1] Control System Toolbox User's Guide ，The MathWorks Inc. 1995

[2] 郑大钟，线性系统理论，清华大学出版社， 1995

[3] 薛定宇，控制系统计算机辅助设计——Matlab 语言及应用，清华大学出版社，1996

[4] 吴麟，自动控制原理，清华大学出版社， 1992

[5] Matlab User's Guide, The MathWorks Inc. 1995

[6] Matlab Reference Guide, The MathWorks Inc. 1995

第 3 章　鲁棒控制工具箱

（Robust Control Toolbox, Ver 2.0.5）

控制理论与应用的发展过程是与人类社会生产的发展与需求紧密联系的。早期的经典控制理论主要解决单输入、单输出的确定性线性系统的控制问题，并在一些简单的工业过程对象中得到应用；其后随着航空航天技术研究的开展，逐渐形成和发展了用于多变量系统的现代控制理论；为克服控制过程中的随机性扰动和参数不确定性，人们提出了以 Kalman 滤波技术为基础的随机控制理论和自适应控制理论（包括模型参考自适应控制和自校正控制）；针对普遍存在的非线性控制对象，非线性控制理论也得到了广泛的研究和应用。上述的控制理论和方法所具有的一个共同缺点是对控制系统模型的精确性要求较高，如经典控制、现代控制和非线性控制方法都要求获得对象的精确数学模型，而随机控制和自校正控制方法要求随机扰动的统计特性满足一定的假设，模型参考自适应控制则主要用于参数未知的情形，且对外界扰动往往不具有鲁棒性。

为克服上述已有的控制理论与方法所存在的共同缺陷，适应人类生产发展中对大量的不确定复杂对象的控制要求，鲁棒控制理论于 20 世纪 80 年代初开始形成和发展，目前已取得了大量的研究成果，成为控制学科的一个重要分支。鲁棒控制理论的研究目标是对存在广泛不确定性的对象提供有效的系统分析和控制器设计的方法，这里所说的广泛不确定性是指对不确定因素的统计特性不作任何假设，而仅认为它属于某个集合，相应的控制对象也因模型的不确定性构成对象族。

鲁棒控制理论经过十多年的发展，取得的主要研究成果有：H_∞控制理论、结构奇异值（μ）分析和综合、频域加权 LQG（H_2控制理论）和 Kharitonov 区间理论等。鲁棒控制方法对控制对象的模型精确性要求不高，且能够有效地克服外界扰动，因而更适于对现实中大量的复杂不确定对象的分析和控制，克服了传统控制理论难以实现工程应用的缺陷。

Matlab 作为一个功能强大的科学计算与工程设计软件，在其工具箱中提供了对各种鲁棒控制分析和设计方法的支持，其中包括鲁棒控制工具箱、μ综合工具箱和线性矩阵不等式 LMI 工具箱。

本章介绍 Matlab 的鲁棒控制工具箱，在该工具箱中提供的鲁棒分析和控制工具有：

（1）　鲁棒性分析工具
- 奇异值
- 特征根轨迹
- 结构奇异值

（2）　鲁棒综合工具
- 频率加权 LQG　（H_2综合）

- LQG/LTR
- H_∞ 综合
- μ 综合工具

（3） 鲁棒模型降阶工具

- 最优描述 Hankel 模型降阶 (具有加性误差界)
- Schur 均衡截断降阶 (具有加性误差界)
- Schur 均衡随机截断降阶（具有乘性误差界）

（4） 采样系统鲁棒控制

3.1 鲁棒控制理论基础

3.1.1 鲁棒控制理论概述

传统的控制理论和方法如经典控制理论、现代控制理论和自适应控制理论等，都要求控制对象的精确模型或要求对象模型的不确定性和外界干扰满足特殊的假定，然而在实际控制系统中，要获得控制对象的精确模型是困难的，甚至是不可能的，对象的不确定性和外界干扰也往往不满足特殊性的假设。因而传统控制理论与实际工程应用之间出现了较大的差距。以 LQG 设计方法为例，该方法是以 Kalman 滤波器和最优二调节理论为基础的反馈设计方法，但 LQG 设计要求获得对象的精确模型，且假定外界干扰信号的统计特征已知。上述条件在实际应用中往往难以达到，所以 LQG 设计方法虽然在理论上具有较好的结果，但其应用却受到很大的限制。

为实现对大量的不确定复杂对象的有效控制，有关学者于 20 世纪 80 年代初开始针对具有一般不确定性和干扰的控制对象进行分析和控制器设计的研究。早期的工作是 Zames 于 1981 年提出的 H_∞ 控制思想，即以从扰动输入到系统输出的传递函数的 H_∞ 范数作为目标函数对系统进行优化设计，从而使扰动对系统输出的影响最小。传递函数的 H_∞ 范数描述了有限输入能量到输出能量的最大增益，因而将其作为控制器优化指标能使具有有限功率谱的干扰对系统的期望输出的影响最小。另外，以 H_∞ 范数作为性能指标还具有如下的优点：（1）可以处理在具有变功率谱的干扰下系统的控制问题；（2） H_∞ 范数具有乘法性质：

$$\|PQ\|_\infty \le \|P\|_\infty \|Q\|_\infty$$

这一性质使其便于研究对象具有不确定性时的鲁棒稳定问题。

H_∞ 控制作为鲁棒控制理论的一个主要分支，在过去的十多年中取得了许多研究成果，其发展过程可以分为如下三个阶段。

第一阶段是从 1981 年到 1984 年，H_∞ 控制理论主要使用逼近方法和插值方法。逼近方法借助于 AAK 理论，在算法上取得了一定的进展，但理论较为复杂；插值方法使用 Nevanlinna-Pick 插值理论以及矩阵形式的 Sarason 理论，具有概念直观的特点，但缺乏有效的算法。1984 年，在 Honeywell 公司举办的 H_∞ 控制研讨会上，Doyle 和 Glover 等对当时的

H_∞ 控制理论进行了总结，从而形成了“1984 年方法”。该方法的框图如图 3.1.1 所示。

在 1984 年方法中涉及的计算有：Youla 参数化、内外分解、谱分解和最佳 Hankel 逼近，其缺点是最后得到的控制器状态数是对象状态数的 10～30 倍，即使采用控制器降阶方法，也只能减为对象状态数的 2～3 倍。

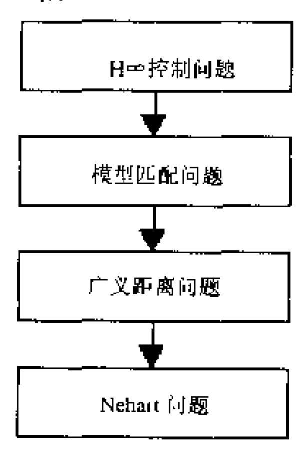


图 3.1.1　H_∞ 控制的“1984 年方法”

在 H_∞ 控制理论发展的第一阶段，H_∞ 控制理论的研究内容还包括混合灵敏度极小优化指标以及具有模型不确定性系统的鲁棒镇定等。

H_∞ 控制理论发展的第二阶段中有代表性的一项工作是标准 H_∞ 控制问题的提出。标准 H_∞ 问题将许多 H_∞ 优化问题包括灵敏度极小化问题、鲁棒镇定问题、混合灵敏度优化问题和模型匹配问题等统一到一个问题框架中，从而使 H_∞ 控制理论的研究更为条理化。另两项具有代表性的工作是 Khargonekar 等人创立的 H_∞ 控制的代数 Riccati 方程解法和基于“2-Riccati 方程”的标准 H_∞ 问题解法。其中“2-Riccati 方程”方法只需求解两个非耦合的代数 Riccati 方程，就可获得阶次不超过广义对象的 McMillan 阶次的 H_∞ 控制器，从而大大简化了 H_∞ 控制问题的求解。

H_∞ 控制理论发展的第三阶段从 1988 年至今，是该理论逐步完善和推广的阶段。有关标准 H_∞ 控制问题解法的研究成果有 Green 等提出的 J-谱分解方法、Kimura 提出的基于 (J,J')-无损分解理论的 H_∞ 控制问题解法以及几种纯时域解法如微分对策方法、极大值原理方法等，其中 (J,J')-无损分解解法是一种具有一定通用性的解法，内外分解、谱分解等都是 (J,J')-无损分解的特例。H_∞ 控制问题的纯时域解法不仅可以解决线性时不变系统的 H_∞ 问题，还可以用来处理时变系统、分布参数系统、非线性系统、奇异摄动系统的 H_∞ 控制问题。在实际控制问题中，除了标准 H_∞ 问题外，还有许多系统不满足标准 H_∞ 问题的条件，即广义对象的传递函数矩阵 $P(s)$ 中的 $P_{12}(s)$ 和 $P_{21}(s)$ 在包含无穷远点在内的虚轴上存在零点，这类系统的 H_∞ 控制问题称为奇异 H_∞ 控制问题或非标准 H_∞ 控制问题。关于奇异 H_∞ 控制问题的解法，目前提出的主要有摄动求解方法、基于二次矩阵不等式的判据和基于 Riccati 不等式的判据、共轭化概念方法以及 (J,J')-无损分解方法在奇异 H_∞ 控制问题中的推广。但上述方法离工程实用的要求仍有一定的差距，有待进一步研究。另外，近年来线性矩阵不等式 LMI 方法在求解 H_∞ 控制问题中应用也取得了一些研究成果。

H_∞ 控制理论作为鲁棒控制理论的重要组成部分，迄今已取得了一系列研究成果，并且在今后仍将成为控制理论与工程界的研究热点。尤其是 H_∞ 控制理论在离散控制系统、非线

性系统中的推广以及在实际工程系统中的应用都将成为重要的研究方向。

在鲁棒控制理论中，结构奇异值（ μ ）分析与综合方法以及 Kharitonov 区间理论是另外的两个重要组成部分。其中 μ 分析与综合方法克服了小增益定理的保守性且能够同时考虑系统的鲁棒稳定性和鲁棒性能，具有广泛的应用。

3.1.2 系统不确定性和鲁棒性

在控制系统分析和设计中，稳定性和动态性能是两项重要的性能指标。在传统控制理论中，系统稳定性和动态性能的分析和设计都是针对模型基本确定的控制对象，但在实际中对象的参数和模型往往具有不确定性，因而限制了传统控制理论的应用。在鲁棒控制理论中，强调对控制系统的鲁棒稳定性和鲁棒性能进行分析。所谓系统的鲁棒性是指当系统的参数在一定范围内变化及一定程度的未建模动态存在时，闭环系统仍能保持稳定性并保证一定的动态性能品质。

研究系统的鲁棒性离不开对系统不确定性的分析。根据对不确定性的假设，系统的不确定性模型可分为随机模型、统计模型、模糊模型和未知有界不确定性模型。在上述不确定性模型中，未知有界不确定性模型具有广泛的代表性，是鲁棒控制的研究对象。

在鲁棒控制理论中，为便于研究，按照未知有界不确定性的特点对其进一步分类。表 3.1.1 表示对系统不确定性的一种分类方法。

表 3.1.1 系统不确定性的分类

<table>
<tr><th colspan="2">类 别</th><th>定义或描述</th></tr>
<tr><td colspan="2">参数不确定性</td><td>$\Gamma=\{G\ (s,q): q\in Q\subset R^n\}$</td></tr>
<tr><td rowspan="3">未建模动态</td><td>加性不确定性</td><td>$G(s,\Delta_A)=G_0(s)+\Delta_A(s)$
$|\Delta_A(j\omega)|\le|W(j\omega)|$</td></tr>
<tr><td>乘性不确定性</td><td>$G(s,\Delta_A)=G_0(s)(1+\Delta_M(s))$
$|\Delta_M(j\omega)|\le|W(j\omega)|$</td></tr>
<tr><td>分子分母不确定性</td><td>$G(s,\Delta_N,\Delta_D)=(N(s)+\Delta_N(s))(D(s)+\Delta_D(s))^{-1}$
标称对象为：$G_0(s)=N(s)D^{-1}(s)$</td></tr>
</table>

在表 3.1.1 中，$G_0(s)$称为标称对象，$\Delta_A(s)$、$\Delta_N(s)$、$\Delta_D(s)$、$\Delta_M(s)$均为不确定性部分对应的传递函数。

在定义系统的不确定性描述后，鲁棒控制的研究对象可由上述不确定性模型给出的一个集合来表示。

相应地，系统的鲁棒性可以由如下若干定义给出。

定义 3.1.1　给定一个不确定性模型集合 Π 和一个性能指标 J，设 G_0 为标称对象模型，$G_0 \in \Pi$，K 为系统的控制器，则闭环系统的鲁棒性定义如下。

(1) 鲁棒稳定性：如果 K 能够对任意 $G \in \Pi$ 保证闭环系统内部稳定，则称闭环系统具有鲁棒稳定性。

(2) 鲁棒性能：如果 K 能够对任意 $G \in \Pi$ 保证闭环系统满足性能指标 J，则称闭环系统具有性能鲁棒性。

3.1.3　控制系统的线性分式变换模型

线性分式变换模型（简称 LFT 模型）能够将许多控制问题用一种统一的标准形式表示出来，因而在 H_∞ 控制、结构奇异值分析等鲁棒控制理论中具有重要的应用价值。下面给出线性分式变换的定义。

定义 3.1.2　记 $M^{m\times n}(s)$ 为所有 $m \times n$ 的有理函数矩阵的全体，设

$$P=\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \in M^{(r_1+r_2)\times(l_1+l_2)},\quad \Delta_l \in M^{l_1\times r_1}(s),\ \Delta_u \in M^{l_2\times r_2}(s)$$

分别定义两个映射

$$F_l(P,\Delta_l): M^{l_1\times r_1}(s) \to M^{r_1\times l_1},$$

$$F_l(P,\Delta_l)=P_{11}+P_{12}\Delta_l(I-P_{22}\Delta_l)^{-1}P_{21}$$

$$F_u(P,\Delta_u): M^{l_2\times r_2}(s) \to M^{r_2\times l_2},$$

$$F_l(P,\Delta_u)=P_{22}+P_{21}\Delta_u(I-P_{22}\Delta_l)^{-1}P_{12}$$

则称 $F_l(P,\Delta_l)$ 为 P 和 Δ_l 的下线性分式变换，$F_u(P,\Delta_u)$ 为 P 和 Δ_u 的上线性分式变换。$F_u(P,\Delta_u)$ 和 $F_l(P,\Delta_l)$ 统称为线性分式变换（LFT：Linear Fractional Transformation）。需要说明的一点是，上述定义是针对有理函数矩阵空间的，对应的是系统的传递函数描述形式；对于复数矩阵空间有类似的定义，对应的是状态空间描述形式。

上线性分式变换和下线性分式变换两种模型的物理意义能够分别以图 3.1.2 和 3.1.3 的形式表达。在图 3.1.2 和图 3.1.3 中，从 $w1$ 到 $z1$ 的闭环传递函数 T_{zw1} 或状态转移矩阵分别为 $F_l(P,\Delta_l)$ 和 $F_u(P,\Delta_u)$，其中 Δ_l 和 Δ_u 为控制器或不确定性部分的传递函数或状态转移矩阵。

在 Matlab 的鲁棒控制工具箱中提供了计算状态空间形式的线性分式变换的函数 lftf，该函数的说明请参见本章的 3.2.3 节——模型转换工具。

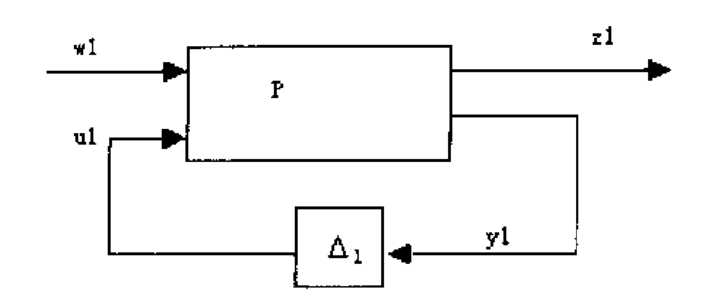


图 3.1.2 下线性分式变换

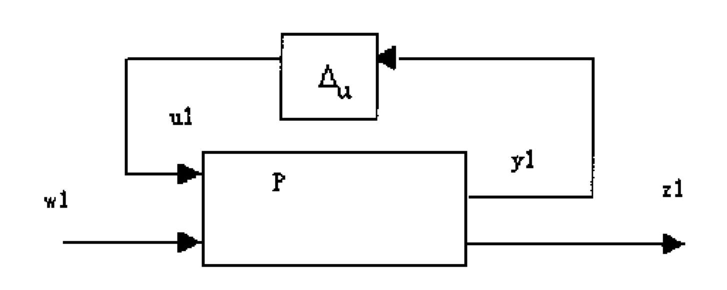


图 3.1.3 上线性分式变换

3.1.4 奇异值与 H_∞、H_2 范数

1 矩阵的奇异值

当系统中的不确定性用一个范数有界的摄动块来表示时，系统的鲁棒稳定性可以根据小增益定理来判断。当标称系统和扰动具有稳定的传递函数时，用矩阵奇异值给出的系统鲁棒性是无保守性的。所以，以奇异值为系统指标的系统设计方法在 20 世纪 70 年代末 80 年代初得到了广泛的研究，成为多变量系统设计的一种重要手段。矩阵的奇异值和奇异值分解定义如下。

定义 3.1.3 设 r 阶矩阵 $A\in C^{m\times n}$, 半正定矩阵 A^*A 的 $p(p=\min\{m,n\})$个特征值为λ_i $(i=1,2,...p)$，则

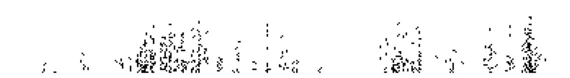
$$\sigma_i=\sqrt{\lambda_i}\qquad(i=1,2,...p)$$

称为矩阵 A 的奇异值。矩阵 A 的最大奇异值和最小奇异值分别记为$\bar{\sigma}(A)$和$\underline{\sigma}(A)$。

定义 3.1.4 设存在两个酉矩阵 $U\in C^{m\times n}$， $V\in C^{m\times n}$和一个 r阶对角矩阵$\Sigma\in C^{m\times n}$满足：

$$A=U\Sigma V^*$$

其中$\Sigma=diag(\sigma_1,\ \sigma_2,\ ...\sigma_n,\ 0,0,...0)$，则上式称为矩阵 A 的奇异值分解。



2　传递函数矩阵的 H_∞ 与 H_2 范数

在鲁棒控制理论中，鲁棒稳定性和鲁棒性能指标通常由传递函数矩阵的 H_∞ 和 H_2 范数描述。为此，在定义矩阵的奇异值之后，给出以奇异值描述的传递函数矩阵的 H_∞、H_2 范数如下。

定义 3.1.5　设 $G(s)$为稳定的 $m\times n$ 维有理函数矩阵，$p=\min\{m, n\}$，则 $G(j\omega)$的 H_∞ 范数为

$$\|G(j\omega)\|_\infty=\sup_\omega[\bar{\sigma}(G(j\omega))]$$

$G(j\omega)$的 H_2 范数为

$$\|G(j\omega)\|_2=[\int_{-\infty}^{\infty}\sum_{i=1}^{p}(\sigma_i(G(j\omega))^2 d\omega]^{\frac{1}{2}}$$

3.1.5　结构奇异值

系统在具有未建模动态时，奇异值分析获得的结果往往较为保守。为克服这一问题，Doyle 于 1982 年提出了结构奇异值的概念，并引起鲁棒控制界的普遍注意。经过 10 多年的研究，结构奇异值方法成为多变量系统的鲁棒稳定性和鲁棒性能分析与设计的重要工具。

对于具有如图 3.1.4 所示一般结构的不确定 M - Δ型系统

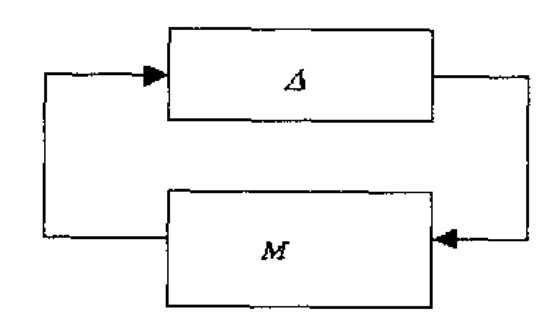


图 3.1.4　M - Δ型不确定系统

Δ为系统的不确定部分，$\Delta=diag(\Delta_1, \Delta_2, \ldots\Delta_n)$。该系统的结构奇异值定义为

$$\mu_\Delta(M):=\begin{cases}(\min\{\bar{\sigma}(\Delta):\det(I-M\Delta)=0,\Delta\in\tilde{\Delta}\})^{-1}\\0,\ \det(I-M\Delta)\neq 0\end{cases}$$

基于结构奇异值的多变量系统鲁棒稳定性定理如下。

定理 3.1　对于图 3.1.4 所示的不确定系统，若标称系统 M 是稳定的，则该闭环系统鲁棒稳定的充要条件是

$$\mu_c=\sup_{\omega\in R}\mu_\Delta(M(j\omega))<1$$

3.2 系统模型建立与转换工具

3.2.1 控制系统模型的数据结构

在 Matlab 鲁棒控制工具箱中，为有效地描述复杂系统的模型，定义了系统模型的树型数据结构及其相关的操作。系统模型可用一个 tree 类型的 Matlab 变量表达，tree 类型的变量具有树型的数据结构，可用 Matlab 的 tree 命令生成。

1 tree

功能：将多个向量和矩阵的数据集成到一个树型结构变量中。

格式：T = tree(nm,b1,b2,...bn)

说明：该函数生成一个树型结构变量T，T中包含了所有分支bi(i=1,2,...n)的数据和维数信息。输入参数bi(i=1,2,...n)构成T的各个分支，bi可以是Matlab的向量或矩阵变量，也可以是字符串变量以及树型结构变量。参数nm为一个包括n个元素的向量，每个元素均为字符串变量，第i个元素作为T的分支bi的名称，nm的各个元素必须用逗号隔开。

另一个与 tree 函数直接相关的函数是 istree。

2 istree

功能：判断一个变量是否为树型变量。

格式：[i] = istree(T)

[i,b] = istree(T,path)

说明：当只有一个参数T时，函数用于判断一个变量是否为树型变量，输入参数T为待判断的变量；当指定参数PATH时，函数用于判断T的某一分支是否存在，该分支由参数path指定。

在第1种情况，若T为一个树型变量，则函数返回值i=1，否则i=0;在第2种情况，若由path指定的T的分支存在，则返回值i=1,参数b为指定的T的分支。

举例：

```
tree1 = tree('a,b,c',a,b,c);
tree3 = tree('w,x',w,x);
tree2 = tree('tree3,y,z',tree3,y,z);
bigtree = tree('tree1,tree2',tree1,tree2);
```

上述语句生成的树型变量 bigtree 的结构如图 3.2.1 所示。

鲁棒控制工具箱还提供了对上述树型变量进行操作的有关函数 branch、graft。

3 branch

功能：从树型变量中返回某一分支元素。

格式：[b1,b2,...bn]=branch(tr,PATH1,PATH2,...PATHn)

说明：输入参数 tr 为树型变量，PATHi(i=1,2,...n)为 tr 的 n 个分支路径名称；输出参数 bi(i=1,2,...n)为 PATHi 对应的分支变量。分支路径名称为一个如下形式的字符串：

PATHi='/name1/name2/.../namen'

其中 name1、name2、...namen 为树型变量 tr 从根节点到某一分支节点的路径上各个节点的字符串名称。

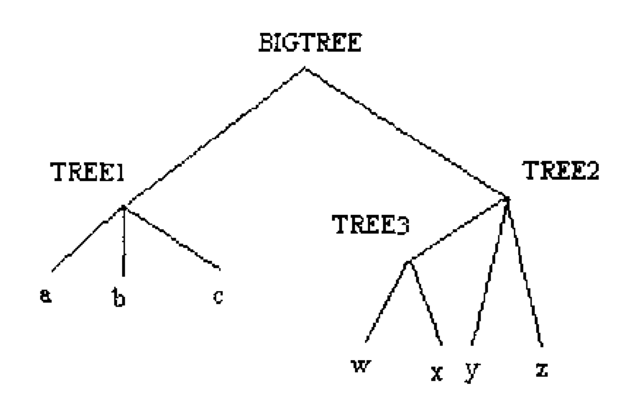


图 3.2.1　树型结构变量

举例：
```
S=tree('a,b,c','foo',[49 50],'bar')
b1=branch(S,'c')
b1='bar'
```

4　graft

功能：添加一个根分支节点到一个树型变量中。

格式：tr1=graft(tr,b)

tr1=graft(tr,b,nm)

说明：输入参数 tr 为一个树型变量，b 为任意类型的变量；输出参数 tr1 为一个新的树型变量，该变量的根节点除包括 tr 的所有根节点外，还包括添加的根节点 b。输入参数 nm 为字符串变量，用于指定新添加的根节点的名称。

举例：
```
S=tree('a,b,c','foo',[49 50],'bar')
tr1=graft(S,'t','new ')
b1=branch(tr1,'new ')
b1='t'
```

在定义了树型变量的基础上，鲁棒控制工具箱提供了 mksys 命令以实现用一个变量来表示控制系统的模型。mksys 命令的作用是将系统的各个组成部分定义的矩阵用一个树型变量来存储，该变量就可表示整个控制系统。

5　mksys

功能：生成表示控制系统模型的变量。

格式：S=mksys(a,b,c,d)

S=mksys(v1,v2,v3,...vn,TY)

说明：函数有两种调用格式。第一种格式有四个固定参数，对应的系统模型为标准的状态空间描述形式，即

$$\dot{x} = ax + bu$$
$$y = cx + du$$

第二种调用格式则具有可变参数个数，对应多种系统模型描述，参数 TY 用于指定系统模型的形式，表 3.2.1 列出了不同的 TY 定义的系统模型和参数格式。

表 3.2.1 系统模型与参数格式

TY 值	参数 vi 的格式	系统模型
'ss'	(a,b,c,d)	标准状态空间模型
'des'	(a,b,c,d,e)	描述系统
'tss'	(a,b1,b2,c1,c2,d11,d12,d21,d22)	两端口状态空间模型
'tdes'	(a,b1,b2,c1,c2,d11,d12,d21,d22,e)	两端口描述了系统
'gss'	(sm,dimx,dimu,dimy,ty)	一般状态空间模型
'gdes'	(e,sm,dimx,dimu,dimy,ty)	一般描述系统
'gpsm'	(psm,deg,dimx,dimu,dimy,ty)	一般多项式系统矩阵
'tf'	(num,den,ty)	传递函数模型
'tfm'	(num,den,m,n,ty)	传递函数矩阵模型
'imp'	(y,ts,nu,ny)	脉冲响应模型

在表 3.2.1 中的系统模型表示中，传递函数模型具有如下的形式：

$$\frac{y(s)}{u(s)} = \frac{num(s)}{den(s)}$$

其中，多项式 $num(s)$和 $den(s)$的各项系数由参数向量 num 和 den 的各个分量决定。

描述系统是状态空间模型的推广，其定义如下。

定义 3.2.1 所谓描述系统，是指可以表示为如下形式的系统：

$$\begin{cases} E\dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

其中，$E \in R^{n\times n}$，$A \in R^{n\times n}$，$B \in R^{n\times q}$，$C \in R^{m\times n}$，$D \in R^{m\times q}$。若矩阵 E 为非奇异矩阵，则描述系统可用状态空间模型表示。

两端口状态空间模型具有如下的形式：

$$\begin{cases} \dot{x} = ax + b_1u_1 + b_2u_2 \\ y_1 = c_1x + d_{11}u_1 + d_{12}u_2 \\ y_2 = c_2x + d_{21}u_1 + d_{22}u_2 \end{cases}$$

相应地，两端口描述系统具有如下的形式：

$$\begin{cases} e\dot{x} = ax + b_1u_1 + b_2u_2 \\ y_1 = c_1x + d_{11}u_1 + d_{12}u_2 \\ y_2 = c_2x + d_{21}u_1 + d_{22}u_2 \end{cases}$$

在表 3.2.1 中，其他参数的定义如下：

dimx,dimy,dimu 分别为状态向量、输出向量和输入向量的维数；

m,n 分别为传递函数分子和分母多项式的次数；

y 为脉冲响应序列。

举例：%生成状态空间模型的系统变量

```
a=[1,3;2,1]
b=[1 1]
c=[2,1;4,3]
d=[4,2]
S1=mksys(a,b,c,d,'ss')
%生成传递函数模型的系统变量
num=[1 2 1]
den=[1 2 3 1]
S2=mksys(num,den,'tf')
%生成描述系统变量
e=[2 4 ;5 7]
a=[1 2;1 8]
b=[2 4]
c=[0.2 1.1 ;2 5]
d=[1 2]
S3=mksys(a,b,c,d,e,'des')
```

与 mksys 函数有关的两个鲁棒控制工具箱函数为 vrsys 和 issys。vrsys 用于返回树型系统模型变量中具有指定模型类型的分支矩阵或向量变量名称，模型类型和分支变量名称由表 3.2.1 定义；issys 用于判断某一变量是否为通过 mksys 函数生成的树型系统模型变量，若是系统模型变量，则返回该变量的分支变量个数。

6　vrsys

功能：返回树型系统模型变量中具有指定模型类型的分支矩阵或向量变量名称。

格式：[VARS,N] = vrsys(NAM)

说明：输入参数 NAM 为具有如下形式的字符串变量：

NAM=[TY'_'SUF]

其中 TY 为模型类型名称，由表 3.2.1 定义；字符串 SUF 为添加到返回变量名称的后缀；返回参数 VARS 为包含相应变量名称的字符串，各个变量名称用逗号隔开；N 为变量的个数。

举例：

```
[vars,n]=vrsys('ss_g')
```

输出：

```
vars=
     ag,bg,cg,dg
n=
          4
```

7 issys

功能：判断某一变量是否为通过 mksys 函数生成的树型系统模型变量。

格式：[I,TY,N] = issystem(S)

说明：输入参数 S 为某一变量名称；当 S 为通过 mksys 函数生成的树型系统模型变量时，输出参数 I＝1,TY 为变量的模型类型，N 为分支变量个数；其他情况 I＝0，TY 为空，n=0。

3.2.2 模型建立工具

除 mksys 函数外，在鲁棒控制工具箱中还提供了三个函数用于建立复杂的控制系统模型，这三个函数为 augss、augtf 和 interc。表 3.2.2 列出了这三个模型建立函数的功能。

表 3.2.2 模型建立工具

函数名称	功能
augss	建立具有加权状态空间矩阵的增广系统模型
augtf	建立具有加权传递函数矩阵的增广系统模型
interc	建立一般的多变量互联系统模型

函数 augss 和 augtf 生成的增广系统模型用于具有加权混合灵敏度的 H_∞ 和 H_2 控制器设计，有关控制器的设计可参见 3.4 节的内容。设开环系统传递函数为 $G(s)$，控制器为 $F(s)$，加权传递函数矩阵分别为 $W_1(s)$、$W_2(s)$和 $W_3(s)$，则闭环传递函数的加权灵敏度矩阵为：

$$T_{y_1u_1}=\begin{bmatrix}W_1S\\W_2R\\W_3T\end{bmatrix}$$

其中，灵敏度矩阵 S、R 和 T 分别定义如下：

$$S=(I+GF)^{-1}$$

$$R=F(I+GF)^{-1}$$

$$T=GF\ (I+GF)^{-1}$$

传递函数矩阵 $G(s)$、$W_1(s)$和 $W_3(s)G(s)$必须为真的，但 $W_3(s)$可以为非真的传递函数矩阵，其表达式如下：

$$W_3(s)=C_{w3}(sI-A_{w3})^{-1}B_{w1}+D_{w1}+P_ns^n+\ldots+P_1S+P_0$$

与具有加权灵敏度传递函数矩阵 $W_1(s)$、$W_2(s)$和 $W_3(s)$的开环系统等价的增广系统可以通过函数 augss 计算得到，该增广系统具有如下的传递函数矩阵：

$$P(s)=\begin{bmatrix} W_1 & -W_1G \\ 0 & W_2 \\ 0 & W_3G \\ I & -G \end{bmatrix}$$

1 augss

功能：建立具有灵敏度加权状态空间矩阵的增广系统模型。

格式：[a,b1,b2,c1,c2,d11,d12,d21,d22] =
augss(ag,bg,cg,dg,aw1,bw1,cw1,dw1,aw2,bw2,cw2,dw2,aw3,bw3,cw3,dw3)
[a,b1,b2,c1,c2,d11,d12,d21,d22]=
augss(ag,bg,cg,dg,aw1,bw1,cw1,dw1,aw2,bw2,cw2,dw2,aw3,bw3,cw3,dw3,
w3poly)
[tss] = augss(ssg,ssw1,ssw2,ssw3,w3poly)
[tss] = augss(ssg,ssw1,ssw2,ssw3)

说明：输入参数 ag、bg、cg 和 dg 为开环系统的状态空间矩阵；awi、bwi、cwi 和 dwi(i=1,2,3)分别为加权灵敏度传递函数矩阵 $W_i(s)$对应的状态空间实现；w3poly 为传递函数 $W_3(s)$的非真多项式部分；ssg、ssw1、ssw2 和 ssw3 为状态空间形式的系统模型变量。输出参数为双端口增广系统的状态空间矩阵或树型变量。

2 augtf

功能：建立具有灵敏度加权传递函数矩阵的增广系统模型。

格式：[a,b1,b2,c1,c2,d11,d12,d21,d22] =augtf(ag,bg,cg,dg,w1,w2,w3)
[tss] = augtf(ssg,w1,w2,w3)

说明：输入参数 ag,bg,cg,dg 为开环系统 G 的状态空间矩阵，ssg 为相应的系统模型变量，w1,w2,w3 为灵敏度加权传递函数矩阵；输出参数为增广系统的状态空间矩阵或相应的树型系统模型变量 tss。

函数 interc 提供了构造一般的多变量互联系统模型的工具，一个典型的多变量互联系统如图 3.2.2 所示。

闭环系统的状态空间实现为：

$$\begin{bmatrix} A_{cl} & B_{cl} \\ C_{cl} & D_{cl} \end{bmatrix}=\begin{bmatrix} A+BFXC & B(M+FXDM) \\ NXC & NXDM \end{bmatrix}$$

其中，A，B，C，D 为 $P(s)$的状态空间实现，X 定义为：

$$X=(I-DF)^{-1}$$

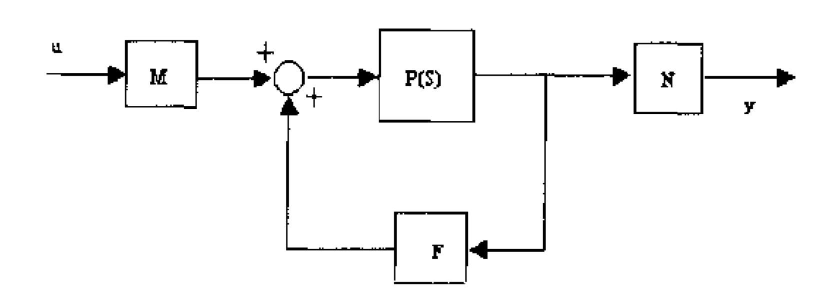


图 3.2.2 多变量互联系统

3 interc

功能：建立多变量互联系统的状态空间模型。

格式：[acl,bcl,ccl,dcl] = interc(a,b,c,d,m,n,f)

[sscl] = interc(ss,m,n,f)

说明：输入参数中，a,b,c,d 为图 3.2.2 中 *P(s)*对应的状态空间矩阵，m,n 和 f 为表示互联关系的常数矩阵。输出参数 ac1,bc1,cc1,dc1 为闭环系统的状态空间矩阵，ssc1 为树型状态空间模型变量。

举例：设有如图 3.2.3 所示的互联系统，*Gi* 对应的状态空间实现设为 *Ai,Bi,Ci,Di*（*i*=1,2,3），首先利用如下的 Matlab 命令生成一个增广对角系统 *P(s)*：

```
[AA,BB,CC,DD] = append(A1,B1,C1,D1,A2,B2,C2,D2);
[AA,BB,CC,DD] = append(AA,BB,CC,DD,A3,B3,C3,D3);
```

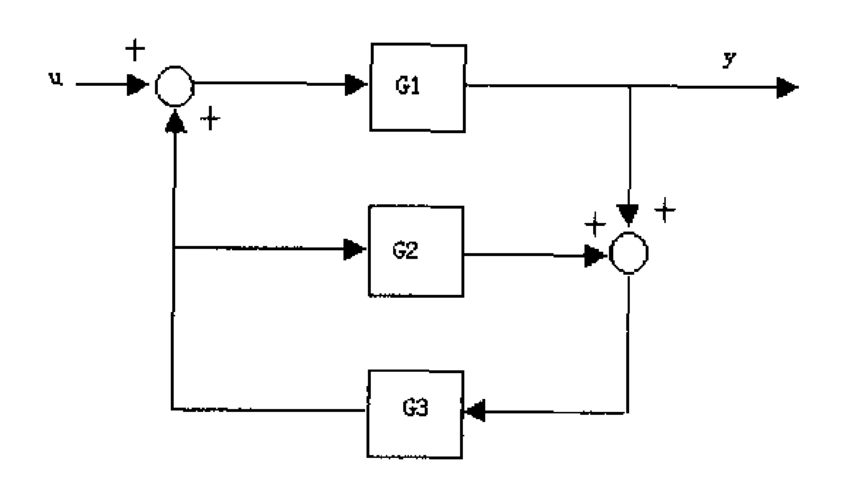


图 3.2.3 互联系统

系统 *P(s)*为：

$$P(s)=\begin{bmatrix} G_1(s) & 0 & 0 \\ 0 & G_2(s) & 0 \\ 0 & 0 & G_3(s) \end{bmatrix}$$

为得到图 3.2.3 中闭环系统的状态空间实现，构造如下的互联关系矩阵：

$$M=\begin{bmatrix} I \\ 0 \\ 0 \end{bmatrix}$$

$$N=[I \quad 0 \quad 0]$$

$$F=\begin{bmatrix} 0 & 0 & I \\ 0 & 0 & I \\ I & I & 0 \end{bmatrix}$$

则利用 interc 函数即可得到闭环系统的状态空间实现

```
[Acl,Bcl,Ccl,Dcl] = interc(AA,BB,CC,DD,M,N,F);
```

3.2.3　模型转换工具

Matlab 鲁棒控制工具箱提供了七个模型转换函数，如表 3.2.3 所示。

表 3.2.3　模型转换函数

函数名称	功　能
bilin	频域多变量双线性变换
des2ss	将描述系统模型转换为状态空间模型
lftf	线性分式变换
sectf	扇区变换
stabproj	稳定/不稳定投影
slowfast	慢速/快速模态分解
tfm2ss	将传递函数矩阵转换为状态空间块控制器形式

1　频域多变量双线性变换

频域双线性变换具有如下的一般形式：

$$s=\frac{\alpha z+\delta}{\gamma z+\beta}$$

其中，α、β、δ 和γ为常数，s 和 z 为复频域变量。该变换主要用于采样控制系统的分析和设计。

鲁棒控制工具箱的 bilin 函数能够计算多变量的频域双线性变换。

- bilin

功能：计算多变量频域双线性变换。

格式：[ab,bb,cb,db] = bilin(a,b,c,d,ver,Type,aug)

[ssb] = bilin(ss,ver,Type,aug)

说明：输入参数 a,b,c,d 为连续系统的状态空间矩阵，ss 为树型状态空间模型变量，参数 ver 用于指定变换的方向即由 $s \to z$（正变换）或由 $z \to s$（逆变换）。当 ver=1 时，进行正变换；当 ver= - 1 时，进行逆变换。

参数 Type 为字符串变量，用于指定变换的类型，aug 为对应不同变换类型的参数列表。函数 bilin 支持 7 种双线性变换，这七种双线性变换的形式和对应的 Type 值如下。

1）Tustin 变换：　Type='Tustin'

$$s=\frac{2}{T}(\frac{z-1}{z+1})$$

参数 aug=T，T 为采样周期。

2）Type='P_Tust'

$$s=\frac{\omega_0}{\tan((\omega_0 T)/2)}(\frac{z-1}{z+1})$$

参数 aug=[T ω_0]。

3）Type='BwdRec',

$$s=\frac{z-1}{Tz}$$

参数 aug=T。

4）Type='FwdRec'

$$s=\frac{z-1}{T}$$

参数 aug=T。

5）Type='S_Tust',

$$s=\frac{2}{T}(\frac{z-1}{\frac{z}{h}+1})$$

参数 aug=[T h]。

6）Type='S-ftjw',

$$s=\frac{z+p_1}{\frac{z}{p_2}+1}$$

参数 aug=[p2 p1]。

7）Type='G_Bilin'，一般的双线性变换

$$s=\frac{\alpha z+\delta}{\gamma z+\beta}$$

参数 aug=[α　β　δ　γ]。

举例：考虑具有如下状态空间实现的连续时间系统：

$$A=\begin{bmatrix}-1 & 1\\ 0 & -2\end{bmatrix}, \quad B=\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$$

$$C=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}, \quad D=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$$

下面的 Matlab 语句对上述系统进行四种形式的双线性变换，并绘出相应的离散奇异值波特图（关于离散奇异值波特图参见 3.3 节）。

```
a=[-1 1;0 -2];
b=[1 0;0 1];
c=[1 0;0 1];
d=[0 0;0 0];
ss = mksys(a,b,c,d);
[sst] = bilin(ss,1,'Tustin',0.05);
[ssp] = bilin(ss,1,'P_Tust',[0.05 40]);
[ssb] = bilin(ss,1,'BwdRec',0.05);
[ssf] = bilin(ss,1,'FwdRec',0.05);
w = logspace(-2,3,100) ;
svt = dsigma(sst,0.05,w);
svp = dsigma(ssp,0.05,w);
svb = dsigma(ssb,0.05,w);
svf = dsigma(ssf,0.05,w);
loglog(w,svt,'.-');
hold on;
loglog(w,svp,'--');
loglog(w,svb,'*');
loglog(w,svf,'-')
```

对应不同类型双线性变换的离散奇异值波特图如图 3.2.4 所示。

2　描述系统向状态空间模型的转换

描述系统模型和状态空间模型是常用的两种系统模型形式。描述系统具有如下的一般表达式：

$$\begin{cases}E\dot{x}=Ax+Bu\\ y=Cx+Du\end{cases}$$

当 E 为非奇异矩阵时，可以方便地将描述系统模型变换为状态空间模型形式；当 E 为奇异矩阵，即 rank(E)<n (n 为状态变量维数)时，描述系统模型和状态空间模型之间的转换

就较为困难。通常采用的一种方法是奇异值分解方法，即对矩阵 E 进行如下的奇异值（SVD: Singular Value Decomposition）分解

$$E=U\begin{bmatrix}\Sigma & 0\\0 & 0\end{bmatrix}V^T=\begin{bmatrix}U_1 & U_2\end{bmatrix}\begin{bmatrix}\Sigma & 0\\0 & 0\end{bmatrix}\begin{bmatrix}V_1^T\\V_2^T\end{bmatrix}$$

其中，$\Sigma=diag(\sigma_1, \sigma_2, \ldots\sigma_r,0,0,\ldots0)$，$U$ 和 V 均为酉矩阵。

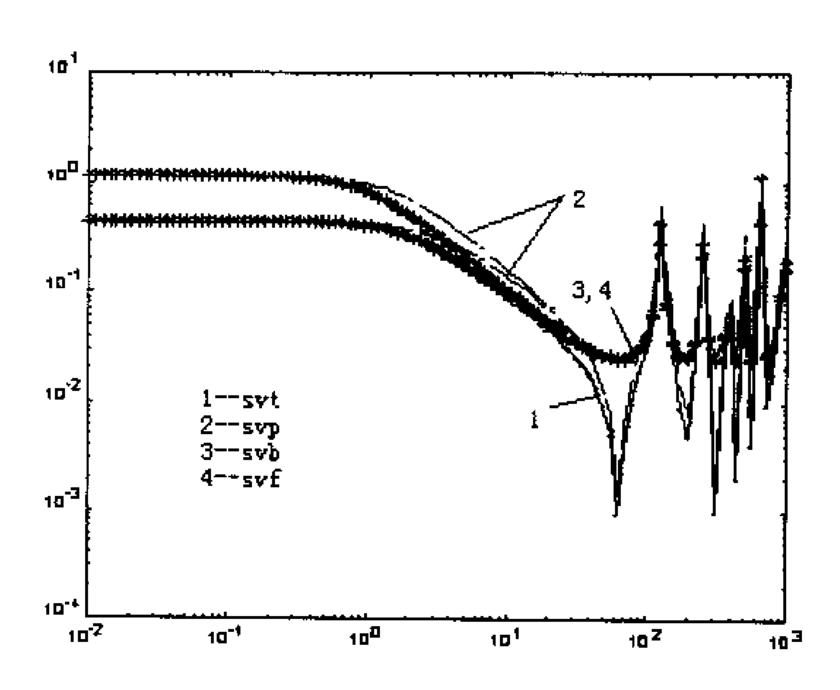


图 3.2.4　不同类型双线性变换的离散奇异值波特图

在上述奇异值分解的基础上，可以对描述系统模型进行状态空间模型的转换。设描述系统 G_1 具有如下的系数矩阵：

$$G_1:\ =\begin{bmatrix}-Es+A & B\\C & D\end{bmatrix}$$

则基于奇异值分解的状态空间模型转换得到下面的结果：

$$\begin{bmatrix}-Is+\Sigma^{-1/2}(A_{11}-A_{12}A_{22}^{-1}A_{21})\Sigma^{-1/2} & \Sigma^{-1/2}(B_1-A_{12}A_{22}^{-1}B_2\\(C_1-C_2A_{22}^{-1}A_{21})\Sigma^{-1/2} & D-C_2A_{22}^{-1}B_2\end{bmatrix}$$

其中，A_{ij}、B_i、$C_i\,(1\le i\le j\le 2)$的定义如下：

$$\begin{bmatrix}A_{11} & A_{12}\\A_{21} & A_{22}\end{bmatrix}=\begin{bmatrix}U_1^T\\U_2^T\end{bmatrix}A\begin{bmatrix}V_1 & V_2\end{bmatrix}$$

$$\begin{bmatrix}B_1\\B_2\end{bmatrix}=\begin{bmatrix}U_1^T\\U_2^T\end{bmatrix}B$$

$$[C_1 \quad C_2] = C[V_1 \quad V_2]$$

在鲁棒控制工具箱中的 des2ss 函数用于完成上述转换的功能。

- **des2ss**

功能：将描述系统模型转换为状态空间模型。

格式：[aa,bb,cc,dd] = des2ss(a,b,c,d,E,k)

[ss1] = des2ss(ss,E,k)

说明：输入参数 a,b,c,d,E 为描述系统模型的系数矩阵，ss 为由 mksys 建立的树型系统模型变量；k 为矩阵 E 的零空间的维数。输出参数为状态空间模型的系数矩阵或对应的树型模型变量。

3　线性分式变换

在 3.1 节中介绍了线性分式变换的有关内容，鲁棒控制工具箱中的 lftf 提供了对单端口或双端口系统进行状态空间的线性分式变换的功能。

- **lftf**

功能：单端口或双端口系统进行状态空间的线性分式变换。

格式：[aa,bb1,bb2,cc1,cc2,dd11,dd12,dd21,dd22] =

lftf(A,B1,B2,C1,C2,D11,D12,D21,D22,a,b1,b2,c1,c2,d11,d12,d21,d22)

[aa,bb,cc,dd] =lftf(a,b1,b2,c1,c2,d11,d12,d21,d22,aw,bw,cw,dw)

[aa,bb,cc,dd] =lftf(aw,bw,cw,dw,a,b1,b2,c1,c2,d11,d12,d21,d22)

tss = lftf(tss1,tss2)

ss = lftf(tss1,ss2)

ss = lftf(ss1,tss2)

说明：输入参数为单端口或双端口系统的状态空间矩阵或对应的树型模型变量，输出参数为闭环系统（单端口或双端口）的状态空间矩阵或树型模型变量。

4　扇区变换(Sector Transformation)

在控制系统设计中，有时系统的输入输出性能指标是以扇区形式给出的。为将扇区形式的性能指标转换为等价的 H_∞ 性能指标，在鲁棒控制工具箱中的 sectf 函数提供了这一功能。在给定一个单端口或双端口系统的状态空间模型 F 和输入输出的扇区性能指标后，sectf 函数计算一个单端口或双端口状态空间系统 G，使得在经过线性分式变换后的闭环系统中，G 的输入输出满足扇区性能指标 secg 等价于 F 满足扇区性能指标 secf。

- **sectf**

功能：状态空间扇区线性分式变换。

格式：[ag,bg1,bg2,cg1,cg2,dg11,dg12,dg21,dg22,at,bt1,bt2,...dt21,dt22] =

sectf (af,bf1,...df22,secf,secg)

[ag,bg,cg,dg,at,bt1,...dt21,dt22] = sectf(af,bf,cf,df,secf,secg)

[tssg,tsst] = sectf (tssf,secf,secg)

```
[ssg,tsst] = sectf (ssf,secf,secg)
```

说明：输入参数的定义如下：

a,bfi,cfi,dfij（i,j=1,2）为双端口系统 F 的状态空间矩阵；

af,bf,cf,df 为单端口系统的状态空间矩阵；

tssf,ssf 为系统的树型模型变量；

secf,secg 分别为系统 F 和 G 的扇区性能指标，它们的取值与对应的性能指 标如表 3.2.4 所示；在表 3.2.4 中，A，B 均为常数矩阵；a,b 为常数向量；S 为分块矩阵，tsss 为双端口系统的树型模型变量。

函数的输出参数定义如下：

ag,bgi,cgi,dgij（i,j=1,2）为双端口系统 F 的状态空间矩阵；

ag,bg,cg,dg 为单端口系统 F 的状态空间矩阵；

at,bti,cti,dtij（i,j=1,2）为系统 F 和 G 的线性分式变换对应的状态空间矩阵；

```
tsst=mksys(at,bt1,bt2,ct1...dt22,'tss')
```

表 3.2.4 扇区性能指标

Secf 或 secg 的取值	扇区性能指标
[-1,1] 或[-1;1]	$\|y\|^2 \le \|u\|^2$
[0,Inf]	$\mathrm{Re}[y^* u] \ge 0$
[A,B]	$\mathrm{Re}[(y-Au)^*(y-Bu)] \le 0$
[a,b]	$\mathrm{Re}[(y-\mathrm{diag}(a)u)^*(y-diag(b)u)] \leqslant 0$
S	$\mathrm{Re}[(S_{12}y+S_{11}u)^*(S_{22}y+S_{21}u)] \leqslant 0$
tsss	$\mathrm{Re}[(S_{12}y+S_{11}u)^*(S_{22}y+S_{21}u)] \leqslant 0$

举例：实现如下的扇区变换并绘制对应系统的 Nyquist 曲线。

$$P(s)=\frac{1}{s+1}\in \sec\mathrm{tor}[-1,1] \to P_1(s)=\frac{s+2}{s}\in \sec\mathrm{tor}[0,\infty]$$

```
[A,B,C,D] = tf2ss(1,[1 1]);
[a,b,c,d] = sectf(A,B,C,D,[-1,1],[0,Inf]);
nyquist(A,B,C,D)
nyquist(a,b,c,d)
```

系统 P 和 P_1 的 Nyquist 图分别如图 3.2.5 和 3.2.6 所示。

5 稳定和不稳定投影、慢速和快速模态分解

所谓稳定和不稳定投影(Stable and Antistable Projections)是指对一个最小实现系统 $G(s)$，将其分为稳定部分和不稳定部分之和，即

$$G(s)=[G(s)]_- +[G(s)]_+$$

其中

$$[G(s)]_{+} = (\hat{A}_{11}, \hat{B}_1, \hat{C}_1, \hat{D}_1)$$

为 $G(s)$的稳定部分，

$$[G(s)]_{-} = (\hat{A}_{22}, \hat{B}_2, \hat{C}_2, \hat{D}_2)$$

为 $G(s)$的不稳定部分。

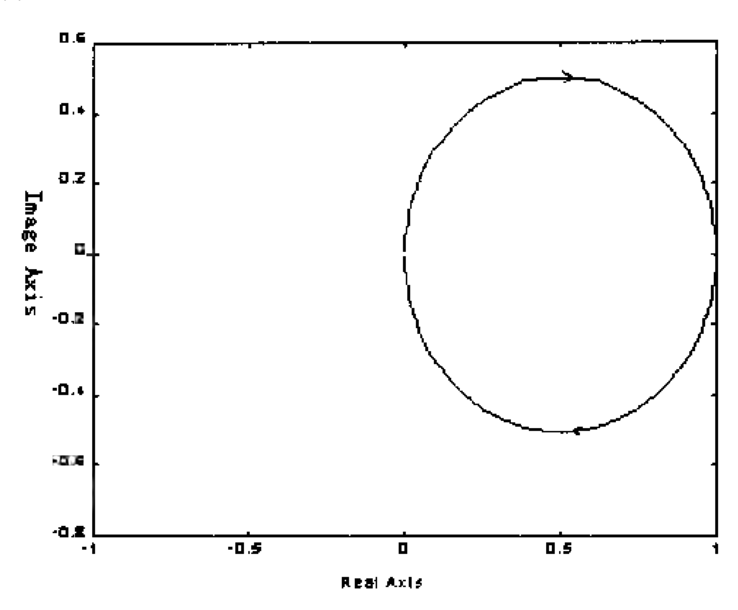


图 3.2.5　系统 $P(s)=\dfrac{1}{s+1}$ 的 Nyquist 图

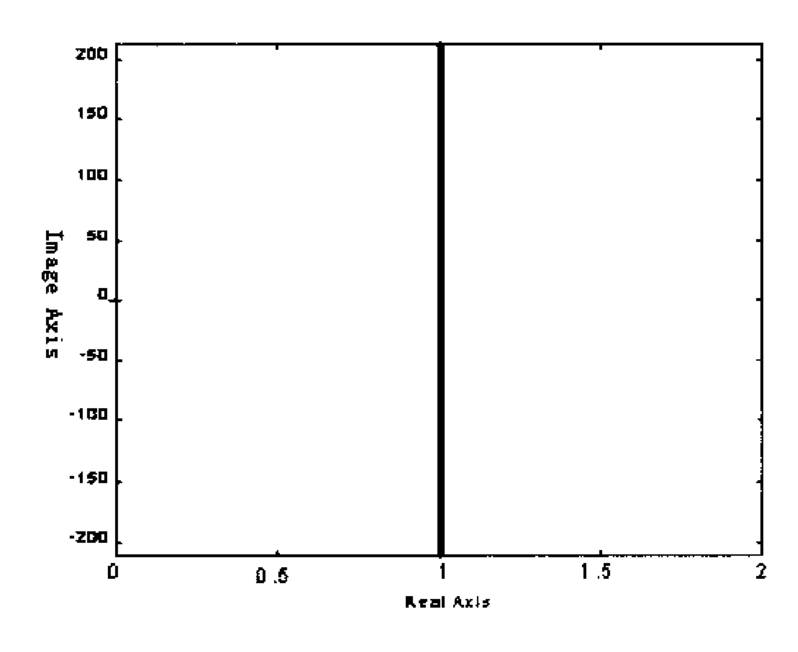


图 3.2.6　系统 $P_1(s)=\dfrac{s+2}{s}$ 的 Nyquist 图

慢速和快速模态分解(Slow and Fast Modes Decompositions) 是指将系统 $G(s)$分为慢速模态对应的传递函数矩阵和快速模态对应的传递函数矩阵之和，即

$$G(s)=[G(s)]_s+[G(s)]_f$$

其中

$$[G(s)]_s:=(\hat{A}_{11},\hat{B}_1,\hat{C}_1,\hat{D}_1)$$

为 $G(s)$的慢速模态部分，

$$[G(s)]_f:=(\hat{A}_{22},\hat{B}_2,\hat{C}_2,\hat{D}_2)$$

为 $G(s)$的快速模态部分。

鲁棒控制工具箱的 stabproj 和 slowfast 函数分别用于完成上述两种系统分解。在这两个函数中采用了如下的算法。

对于系统状态空间模型（A，B，C，D），利用鲁棒控制工具箱的函数 blksch 或 rschur（参见 3.4 节）对矩阵 A 进行有序 Shur 分解，即得到如下结果：

$$\hat{A}=VAV=\begin{bmatrix}\hat{A}_{11} & \hat{A}_{12}\\ 0 & \hat{A}_{22}\end{bmatrix}$$

当采用不同的有序 Schur 分解形式时，分别得到对应于稳定和不稳定投影以及对应于慢速快速模态分解的的 $\hat{A}_{11}$ 和 $\hat{A}_{22}$，其中后者满足$|\lambda_i(\hat{A}_{11})|<|\lambda_i(\hat{A}_{22})|$。然后求解下面的矩阵方程以计算矩阵 X：

$$\hat{A}_{11}X-X\hat{A}_{22}+\hat{A}_{12}=0$$

最后得到状态空间投影：

$$G(s)=[G(s)]_-+[G(s)]_+$$

$$=\begin{bmatrix}\hat{A}_{11} & \hat{B}_1\\ C_1 & 0\end{bmatrix}+\begin{bmatrix}\hat{A}_{22} & \hat{B}_2\\ C_2 & D\end{bmatrix}$$

或

$$G(s)=[G(s)]_s+[G(s)]_f$$

$$=\begin{bmatrix}\hat{A}_{11} & \hat{B}_1\\ C_1 & 0\end{bmatrix}+\begin{bmatrix}\hat{A}_{22} & \hat{B}_2\\ C_2 & D\end{bmatrix}$$

其中

$$\begin{bmatrix}\hat{B}_1\\ \hat{B}_2\end{bmatrix}:=\begin{bmatrix}I & -X\\ 0 & I\end{bmatrix}VB$$

$$[\hat{C}_1 \quad \hat{C}_2]=CV^T\begin{bmatrix}I & X\\ 0 & I\end{bmatrix}$$

函数 stabproj 和 slowfast 的使用说明如下。

- stabproj

  功能：状态空间的稳定和不稳定投影。

  格式：[a1,b1,c1,d1,a2,b2,c2,d2,m] = stabproj(a,b,c,d)

  [ss1,ss2,m] = stabproj(ss)

  说明：输入参数定义如下：

  a,b,c,d——系统的状态空间矩阵；

  ss——系统模型对应的树型模型变量。

  输出参数定义如下：

  a1,b1,c1,d1——系统的稳定状态空间投影，ss1为相应的树型模型变量；

  a1,b2,c2,d2——系统的不稳定状态空间投影，ss2为相应的树型模型变量；

  m为系统矩阵A的稳定特征值的个数。

- slowfast

  功能：系统状态空间的慢速和快速模态分解。

  格式：[a1,b1,c1,d1,a2,b2,c2,d2] = slowfast(a,b,c,d,cut)

  [ss1,ss2] = slowfast(ss,cut)

  说明：输入参数定义如下：

  a,b,c,d——系统的状态空间矩阵；

  ss——系统模型对应的树型模型变量；

  cut——模态分解的指数。

  输出参数定义如下：

  a1,b1,c1,d1——系统的慢速模态，ss1为相应的树型模型变量；

  a1,b2,c2,d2——系统的快速模态，ss2为相应的树型模型变量。

6 将传递函数矩阵转换为状态空间块控制器形式

函数 tfm2ss 用于将传递函数矩阵 $G(s)$转换为状态空间块控制器的形式，其中传递函数矩阵 $G(s)$具有如下的形式：

$$G(s)=\frac{N(s)}{d(s)}$$

上式中，$N(s)$为 $r\times c$ 矩阵，其第 i 行、第 j 列的元素为：

$$[N(\mathrm{s})]_{ij}=\beta_{ij0}s^n+\beta_{ij1}s^{n-1}+\ldots+\beta_{ijn}$$

$d(s)$为 $G(s)$各个元素分母的最小公倍式，具有如下的形式：

$$d(s)=\alpha_0 s^n+\alpha_1 s^{n-1}+\ldots\alpha_n$$

下面给出函数 tfm2ss 的使用说明。

- tfm2ss

  功能：将传递函数矩阵 $G(s)$转换为状态空间块控制器的形式。

格式：[a,b,c,d]=tfm2ss (num,den,r,c)

[a,b,c,d]=tfm2ss (tf,r,c)

说明：输入参数定义为：

num= $[N_{11}\ N_{21}...N_{r1}\ ...N_{rc}]^T$, $N_{ij}=[\beta_{ij0}\ \beta_{ij1}\ ...\beta_{ijn}]$

为 $N(s)$ 各个多项式元素的系数；

den= $[\alpha_0\ \alpha_1\ ..\alpha_n]$ 为 $d(s)$ 的各项系数；

r,c 为 $N(s)$ 的维数；

tf 为由 num 和 den 构成的传递函数矩阵。

输出参数定义为：

a,b,c,d——状态空间块控制器的系数矩阵；

ss——状态空间块控制器对应的系统变量。

3.3 多变量波特图

基于系统频率响应的波特图是分析和设计控制系统的有力工具，在鲁棒控制工具箱中，提供了计算和绘制多变量波特图的有关函数。这些函数能够绘制多变量系统的特征增益波特图、奇异值波特图以及结构化奇异值波特图等多种形式的波特图，如表 3.3.1 所示。

表 3.3.1 鲁棒控制工具箱的多变量波特图函数

函数名称	功能
cgloci	连续系统特征增益/相位波特图
dcgloci	离散系统特征增益/相位波特图
dsigma	离散系统奇异值波特图
muopt	利用乘子尺度化方法计算结构奇异值上界
osborne	基于 Osborne 方法计算结构奇异值上界
perron,psv	Perron 特征结构奇异值
sigma	连续系统奇异值波特图
ssv	结构奇异值波特图

3.3.1 频率响应的特征增益/相位波特图

系统 $G(s)$ 频率响应的特征增益（Characteristic Gain）和相位定义如下：

$$\text{特征增益：}\ g(\omega)=|\mathrm{eig}(G(j\omega))|$$

$$\text{特征相位：}\ \phi(\omega)=\angle\mathrm{eig}(G(j\omega))$$

其中 $g(\omega)$ 和 $\phi(\omega)$ 均为向量，维数等于传递函数矩阵 $G(s)$ 的特征值的个数。

鲁棒控制工具箱的 cgloci 函数和 dcloci 函数分别用于计算和绘制连续和离散多变量系统的特征增益波特图。

1　cgloci

功能：计算和绘制连续多变量系统的特征增益波特图。

格式：[cg,ph,w] = cgloci(a,b,c,d)

[cg,ph,w] = cgloci(a,b,c,d,'inv')

[cg,ph,w] = cgloci(a,b,c,d,w)

[cg,ph,w] = cgloci(a,b,c,d,w,'inv')

[cg,ph,w] = cgloci(ss,...)

说明：输入参数的定义如下：

a,b,c,d——系统状态空间矩阵；

ss——系统状态空间模型变量；

'inv'——指定计算和绘制逆系统 $G^{-1}(s)$ 的特征增益相位波特图；

w——频率向量，用于指定计算特征增益和相位的频率点。

输出参数的定义如下：

cg——系统频率响应的特征增益向量；

ph——系统频率响应的特征相位向量；

w——频率向量。

当不指定输出变量时，函数将绘制系统的特征增益和相位的波特图。

举例：考虑 2×2 的传递函数矩阵

$$G(s)=\begin{bmatrix}\dfrac{-47s+2}{(s+1)(s+2)} & \dfrac{56s}{(s+1)(s+2)}\\ \dfrac{42s}{(s+1)(s+2)} & \dfrac{50s+2}{(s+1)(s+2)}\end{bmatrix}$$

$G(s)$的对角化分解如下

$$G(s)=X\Lambda X^{-1}=\begin{bmatrix}7 & -8\\ -6 & 7\end{bmatrix}\begin{bmatrix}\dfrac{1}{s+1} & 0\\ 0 & \dfrac{2}{s+2}\end{bmatrix}\begin{bmatrix}7 & 8\\ 6 & 7\end{bmatrix}$$

采用下面的 Matlab 语句绘制该系统的特征增益波特图，如图 3.3.1。

```
num=[-47 2;56 0; -42 0;50 2]
den=[1 3 2]
[a,b,c,d]=tfm2ss (num,den,2,2)
cgloci(a,b,c,d)
```

2　dcgloci

功能：计算和绘制离散多变量系统的特征增益波特图。

格式：[cg,ph,w] = dcgloci(a,b,c,d,Ts)

[cg,ph,w] = dcgloci(a,b,c,d,Ts,'inv')

[cg,ph,w] = dcgloci(a,b,c,d,Ts,w)

[cg,ph,w] = dcgloci(a,b,c,d,w,Ts,'inv')

[cg,ph,w] = dcgloci(ss,...)

说明：输入参数的定义如下：

a,b,c,d——系统状态空间矩阵；

ss——系统状态空间模型变量；

'inv'——指定计算和绘制逆系统 $G^{-1}(s)$ 的特征增益相位波特图；

w——频率向量，用于指定计算特征增益和相位的频率点；

Ts——离散系统的采样周期。

输出参数的定义如下：

cg——系统频率响应的特征增益向量；

ph——系统频率响应的特征相位向量；

w——频率向量。

与 cgloci 函数类似，当不指定输出参数时，dcgloci 将绘制出系统的特征增益和相位的波特图。

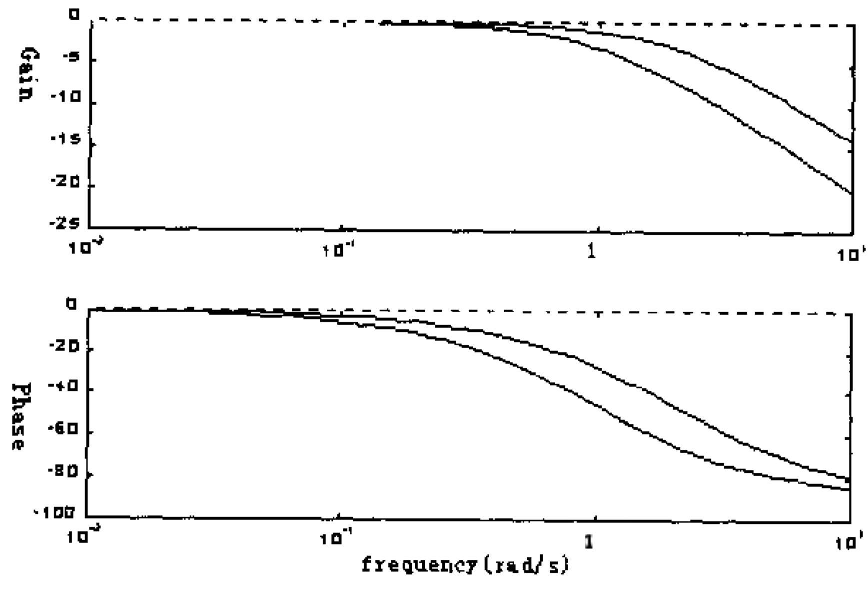


图 3.3.1 系统 $G(s)$的特征增益和相位的波特图

3.3.2 连续和离散系统的奇异值波特图

奇异值波特图是单变量系统频率响应幅值和相位波特图在多变量系统的推广，是分析和设计多变量系统的重要工具。在鲁棒控制工具箱中提供了两个函数分别用于计算和绘制多变量系统的奇异值波特图，即 sigma 和 dsigma。

1 sigma

功能：计算和绘制连续多变量系统的奇异值波特图。

格式：[sv,w] = sigma(a,b,c,d)

```
[sv,w] = sigma(a,b,c,d,'inv')
[sv,w] = sigma(a,b,c,d,w)
[sv,w] = sigma(a,b,c,d,w,'inv')
[sv,w] = sigma(ss,..)
```

说明：输入参数的定义如下：

a,b,c,d——系统状态空间矩阵；

ss——系统状态空间模型变量；

'inv'——指定计算和绘制逆系统 $G^{-1}(s)$ 的特征增益相位波特图；

w——频率向量，用于指定计算特征增益和相位的频率点。

输出参数定义如下：

sv——系统的奇异值矩阵，该矩阵的每一列对应一个奇异值在各个频率点的幅值；

w——频率向量。

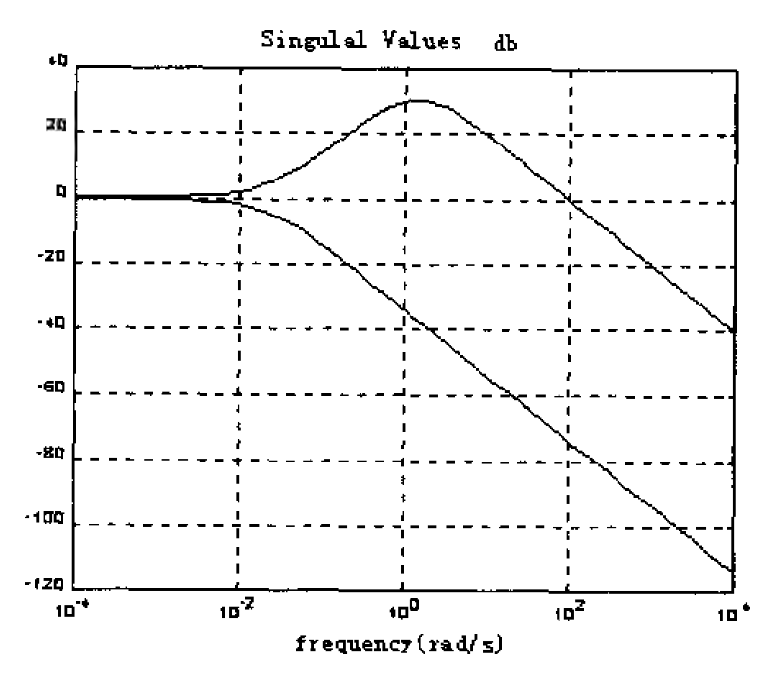


图 3.3.2　连续系统 $G(s)$的奇异值波特图

举例：考虑与函数 cgloci 举例相同的 2×2 系统矩阵

$$G(s)=\begin{bmatrix}\dfrac{-47s+2}{(s+1)(s+2)} & \dfrac{56s}{(s+1)(s+2)}\\ \dfrac{-42s}{(s+1)(s+2)} & \dfrac{50s+2}{(s+1)(s+2)}\end{bmatrix}$$

采用下面的 Matlab 语句计算和绘制 $G(s)$的奇异值波特图，如图 3.3.2 所示。

```
num=[-47 2;56 0; -42 0;50 2]
den=[1 3 2]
```

```
[a,b,c,d]=tfm2ss (num,den,2,2)
w=logspace(-4,4)
sigma(a,b,c,d,w)
```

2 dsigma

功能：离散系统的奇异值波特图。

格式：[sv,w] = dsigma(a,b,c,d,Ts)

[sv,w] = dsigma(a,b,c,d,Ts,'inv')

[sv,w] = dsigma(a,b,c,d,Ts,w)

[sv,w] = dsigma(a,b,c,d,Ts,w,'inv')

[sv,w] = dsigma(ss,..)

说明：在输入输出参数中，Ts 为采样周期；其他参数的定义与 sigma 函数相同。

举例：计算和绘制多变量系统 $G(s)$在采样周期为 0.05s 时的离散奇异值波特图，如图 3.3.3 所示。其中

$$G(s)=\begin{bmatrix}\dfrac{-47s+2}{(s+1)(s+2)} & \dfrac{56s}{(s+1)(s+2)}\\ \dfrac{-42s}{(s+1)(s+2)} & \dfrac{50s+2}{(s+1)(s+2)}\end{bmatrix}$$

```
num=[-47 2;56 0; -42 0;50 2]
den=[1 3 2]
[a,b,c,d]=tfm2ss (num,den,2,2)
w=logspace(-4,4)
dsigma(a,b,c,d,0.05,w)
```

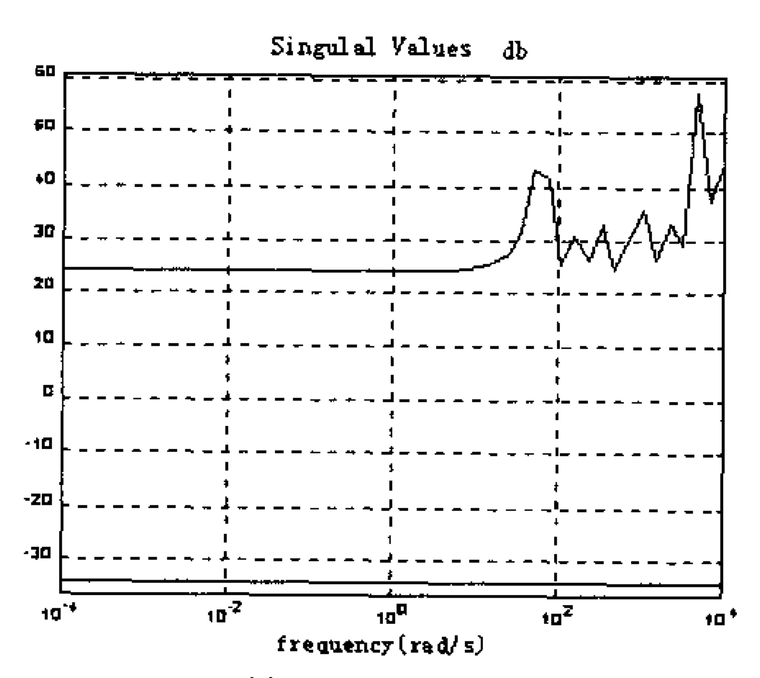


图 3.3.3 离散奇异值波特图

3.3.3　结构奇异值波特图

在利用结构奇异值方法进行系统的鲁棒稳定性分析和设计时，需要对结构奇异值的上界进行计算和估计。目前提出了多种计算结构奇异值上界的方法，主要有 Perron 对角尺度化(Diagonal Scaling)方法、Osborne 对角尺度化方法和乘子尺度化(Multiplier Scaling)方法。

在 Matlab 鲁棒控制工具箱中提供了实现上述计算结构奇异值上界方法的函数，即 muopt、osborne、ssv 和 psv。

1　乘子尺度化方法(Multiplier Scaling)

乘子尺度化方法是基于推广的 Popov 乘子理论。该理论由 Safonov 和 Lee 提出，并利用 Fan 和 Nekooie 的算法。乘子尺度化方法计算得到的具有 n 个不确定块的矩阵 A 的结构奇异值上界为下面的优化问题的解：

$$\min_{M} \ \mu$$

约束条件

$$A_s + A_s^* \geq 0$$

其中，矩阵 A_s 为 A 的 Popov 乘子尺度化矩阵，即

$$A_s = M^{\frac{1}{2}}(\mu I - A)(\mu I + A)^{-1} M^{-\frac{1}{2}*}$$

M 为不确定结构的各个块对角部分对应的广义 Popov 乘子构成的集合。

完成以上功能的 Matlab 函数为 muopt。

- muopt

功能：利用乘子尺度化方法计算结构奇异值上界。

格式：[mu,ascaled,logm,x] = muopt(a)

[mu,ascaled,logm,x] = muopt(a,k)

说明：输入参数 a 为标称矩阵，k 为不确定部分的块结构的维数，k 为二维向量，其第一列为各个块矩阵的行维数，第二列为相应的列维数；如果 k 省略，则缺省值为 k=ones(p,2)，即各个块均为 1×1 的矩阵。

输出参数的定义如下：

mu——矩阵 a 的结构奇异值上界；

ascaled——经过乘子尺度化后的矩阵 A，设乘子尺度化矩阵为 A_s，则经过尺度化后的矩阵 A 定义为：

$$A=\mu\ (I-A_s)(I+A_s)^{-1};$$

logm—— log(diag($M^{\frac{1}{2}}$));

x——矩阵 $A_s + A_s^*$ 的最小特征值的归一化特征向量。

2 Osborne 对角尺度化(Diagonal Scaling)方法

设标称矩阵 A 具有块对角形式的不确定性，不确定块的维数由二维向量 k 表示，其第一列为各个块矩阵的行维数，第二列为相应的列维数。

Osborne 对角尺度化方法采用了如下的算法。

Step 1：将 A 按照其不确定块的维数进行分割；

Step 2：构造 $n \times n$ 的矩阵 F，F 的元素为矩阵 A 的各个块矩阵的最大奇异值，其中 n 为不确定块的个数；

Step 3：计算对角尺度化矩阵 D，D 满足如下极小化指标：

$$\min\left| DFD^{-1} \right|_F ,\quad \left|\cdot\right|_F \text{ 为 Frobenius 范数。}$$

在完成矩阵 A 的 Osborne 对角尺度化的基础上，就可利用对角尺度化矩阵 DAD^{-1} 的最大奇异值对矩阵 A 的结构奇异值的上界进行估计。

函数 Osborne 用于计算矩阵 A 的对角尺度化矩阵和尺度化矩阵的最大奇异值，该函数的说明如下。

· Osborne

功能： Osborne 对角尺度化和结构奇异值上界估计。

格式： [mu,ascaled,logd] = osborne(a)

[mu,ascaled,logd] = osborne(a,k)

说明： 输入参数的定义如下：

a——标称矩阵；

k——不确定部分的块结构的维数，k 为二维向量，其第一列为各个块矩阵的行维数，第二列为相应的列维数；如果 k 省略，则缺省值为 k=ones(p,2)，即各个块均为 1×1 的矩阵。

输出参数的定义为：

mu——对角尺度化矩阵的最大奇异值，即矩阵 a 的结构奇异值上界的估计；

ascaled——尺度化矩阵 DAD^{-1}；

logd——对角尺度化矩阵 D 的对数矩阵。

举例：

例1.

```
A = eye(10);
A(1,10) = 100000;
[mu,Ascaled,logd] = osborne(A);
mu
```

输出结果：

```
mu=
```

```
        1.0000
```

例2.

```
A = eye(8);
A(1,3) = 100000;
A(4,8) = 500000;
[mu,Ascaled,logd] = osborne(A);
mu
```

输出结果：

```
    mu=
      1.0000
```

3　基于 Perron 特征向量方法的结构奇异值上界估计

给定系统标称矩阵 A，以及块结构形式的不确定部分，对矩阵 A 按照不确定块的结构进行分割并构造 $n\times n$ 的非负矩阵 F，F 的元素为矩阵 A 的各个块矩阵的最大奇异值，其中 n 为不确定块的个数；则矩阵 F 的 Perron 特征值定义如下：

$$\lambda_p(F)=\max_i \mathrm{Re}(\lambda_i(F))$$

相应于 Perron 特征值λ_p的左特征向量 y_p 和右特征向量 x_p 称为 F 的 Perron 特征向量。

1982 年，Safonov 分析了对矩阵 A 的结构奇异值上界进行估计的问题，得到了如下的结果：

$$\mu(A)\le\inf_{\mathrm{D}}\left\|DAD^{-1}\right\|_\infty\le\left\|D_pAD_p^{-1}\right\|_\infty\le\lambda_p(F)$$

其中，D 为对角尺度化矩阵，D_p 为 Perron 最优对角尺度化矩阵，D_p 的定义如下：

$$D_{\mathrm{p}}=\mathrm{diag}(d_1,d_2,\dots d_n)\ ,d_i=\sqrt{\frac{y_{pi}}{x_{pi}}}$$

由上述不等式可以看出，采用 Perron 特征向量的对角尺度化方法可以获得对矩阵 A 的结果奇异值上界的较好估计。

鲁棒控制工具箱的 Perron 函数用于完成基于 Perron 特征值的结构奇异值上界计算，函数 psv 用于完成基于 Perron 最优尺度化的结构奇异值上界计算。

- perron

功能：基于 Perron 特征值的结构奇异值上界计算。

格式：[mu] = perron(a)

[mu] = perron(a,k)

说明：输入参数定义为：

a——标称矩阵；

k——不确定部分的块结构的维数，k 为二维向量，其第一列为各个块矩阵的行维数，第二列为相应的列维数；如果 k 省略，则缺省值为 k=ones(p,2)，即各个块均为 1×1 的矩阵。

输出参数定义为：

mu——非负矩阵 F 的 Perron 特征值，其中 F 的元素为 A 的各个块矩阵的最大奇异值。该特征值作为 A 的结构奇异值上界的估计。

- **psv**

功能：基于 Perron 最优尺度化的结构奇异值上界计算。

格式：[mu,ascaled,logd] = psv(a)

[mu,ascaled,logd] = psv(a,k)

说明：输入参数定义为：

a——标称矩阵 A；

k——不确定部分块结构的维数，k 为二维向量，其第一列为各个块矩阵的行维数，第二列为相应的列维数；如果 k 省略，则缺省值为 k=ones(p,2)，即各个块均为 1×1 的矩阵。

输出参数定义为：

mu——A 的 Perron 最优尺度化矩阵的最大奇异值，即

$$\text{mu} = \bar{\sigma}(D_p A D_p)$$

$$D_p = diag(d_1, d_2, \ldots d_n)\ , d_i = \sqrt{\frac{y_{pi}}{x_{pi}}}$$

x_{pi} 和 y_{pi} 分别为 F 的 Perron 特征向量 x_p、y_p 的各个分量。

ascaled——矩阵 A 的 Perron 最优尺度化矩阵；

ogd——D_p 的对数矩阵。

举例：

```
A = eye(10);
A(1,10) = 100000;
[mu,Ascaled,logd] = psv(A);
mu
```

输出结果：

```
mu=
      1.000
```

4　结构奇异值波特图

前面介绍了三种计算结构奇异值上界的方法，基于以上方法，利用函数ssv可以计算和绘制多变量不确定系统的结构奇异值波特图。

- **ssv**

功能：计算和绘制结构奇异值波特图。

格式：[mu,logd] = ssv(a,b,c,d,w)

[mu,logd] = ssv(a,b,c,d,w,k)

[mu,logd] = ssv(a,b,c,d,w,k,opt)

[mu,logd] = ssv(ss,...)

说明：函数ssv能够调用函数osborne、perron、psv和muopt四个函数之一来计算如下的传递函数矩阵的结构奇异值上界并绘制相应的波特图。

$$G(j\omega) = C(j\omega I - A)^{-1}B + D$$

其中矩阵A、B、C、D为系统状态空间矩阵。

输入参数的定义如下：

a,b,c,d——标称系统的状态空间矩阵；

w——频率向量；

k——不确定部分块结构的维数，k 为二维向量，其第一列为各个块矩阵的行维数，第二列为相应的列维数；如果 k 省略，则缺省值为 k=ones(p,2)，即各个块均为 1×1 的矩阵；

opt——选择计算结构奇异值上界的方法，opt 可为下列取值之一：

- 'muopt'——Popov 乘子尺度化方法；
- 'osborne'——对角尺度化方法；
- 'perron'——perron 特征值方法；
- 'psv'——perron 最优对角尺度化方法。

ss——系统状态空间模型变量。

输出参数定义如下：

mu——系统在各个频率点的结构奇异值上界值；

logd——最优对角尺度矩阵 $D(j\omega)$的对数波特图，如果不确定块均为复的，则 $D(j\omega)$为实的；如果不确定块同时具有实参数摄动和复摄动，则 logd 通常为复的，并包括最优乘子尺度的平方根波特图。

3.4　鲁棒控制综合方法

3.4.1　LQG 优化控制综合

设控制对象具有如下的状态空间模型：

$$\begin{cases} \dot{x} = Ax + bu + \xi \\ y = Cx + Du + \theta \end{cases}$$

其中，ξ、θ 为高斯型白噪声，且具有如下的协方差矩阵：

$$E\{\begin{bmatrix} \xi(t) \\ \theta(r) \end{bmatrix}[\xi(t)\,\theta(r)]^T\} = \begin{bmatrix} \Xi & N_f \\ N_f^{\ T} & \Theta \end{bmatrix}\delta(t-r)$$

LQG 优化控制的性能指标为：

$$J_{LQG}=\lim_{T\to\infty}E\{\int_0^T [x^T\ u^T]\begin{bmatrix}Q & N_c\\ N_c^T & R\end{bmatrix}\begin{bmatrix}x\\ u\end{bmatrix}dt\}$$

对应的闭环控制系统的结构如图 3.4.1 所示。

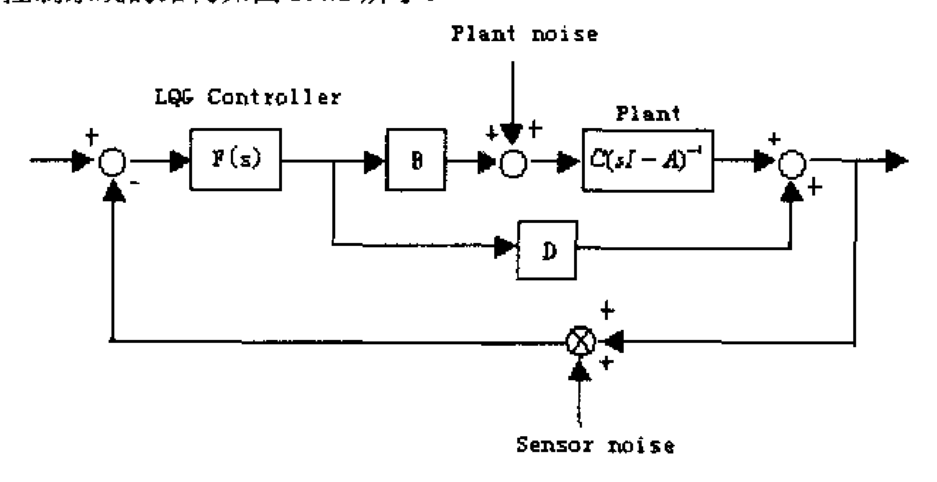


图 3.4.1　LQG 闭环控制系统

根据分离原理，LQG 优化控制问题的解是 Kalman 滤波问题和全状态反馈问题的综合，求解得到的负反馈控制器具有如下的形式：

$$u=-F(s)y$$

$$F(s)=\begin{bmatrix}A-K_fC_2-B_2K_c+K_fD_{22}K_c & K_f\\ K_c & 0\end{bmatrix}$$

函数 lqg 用于计算上述闭环系统的 LQG 优化控制器。

- **lqg**

  功能：计算 LQG 优化控制器。

  格式：[af,bf,cf,df] = lqg(A,B,C,D,W,V)

  [ssf] = lqg(ss,w,v)

  说明：输入参数定义为：

  A，B，C，D——开环系统的状态空间矩阵；

  W——用于指定 LQG 优化指标的加权矩阵

$$W=\begin{bmatrix}Q & N_c\\ N_c^T & R\end{bmatrix}$$

  V——用于噪声的协方差矩阵

$$V=\begin{bmatrix}\Xi & N_f\\ N_f^T & \Omega\end{bmatrix}$$

  输出参数定义如下：

  af,bf,cf,df——反馈控制器的状态空间矩阵；

ssf——反馈控制器的状态空间模型变量。

3.4.2 连续/离散 H_2 综合

1 H_2 优化控制问题的描述

考虑如图 3.4.2 所示的闭环控制系统，其中 $F(s)$ 为控制器，$P(s)$ 为两端口的控制对象，其增广状态空间矩阵定义为：

$$P=\begin{bmatrix} A & B_1 & B_2 \\ C_{11} & D_{11} & D_{12} \\ C_{21} & D_{21} & D_2 \end{bmatrix}$$

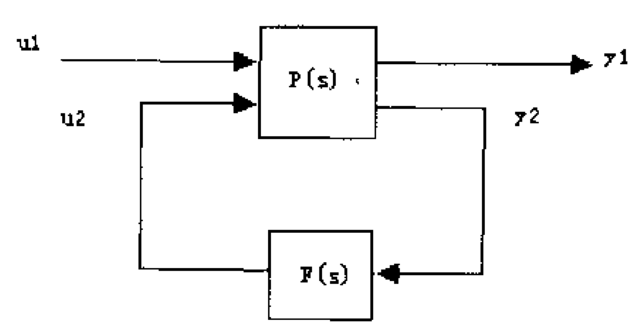


图 3.4.2 闭环控制系统

H_2 优化控制器设计的性能指标为：

$$\min_{F(s)}\left\|T_{y_1u_1}\right\|_2=\min_{F(s)}\left(\frac{1}{\pi}\int_0^\infty trace(T^*{}_{y_1u_1}(j\omega))d\omega\right)^{1/2}$$

求解上述 H_2 优化控制问题的方法是将其转化为等价的 LQG 优化控制问题，与 H_2 优化指标等价的 LQG 优化指标为：

$$\begin{aligned} J_{LQG} &= \lim_{T\to\infty} E\left\{\frac{1}{T}\int_0^T y_1^T y_1 dt\right\} \\ &= \lim_{T\to\infty} E\left\{\frac{1}{T}\int_0^T [x^T \quad u_2^T]\begin{bmatrix} Q & N_c \\ N_c^T & R \end{bmatrix}\begin{bmatrix} x \\ u_2 \end{bmatrix}dt\right\} \\ &= \lim_{T\to\infty} E\left\{\frac{1}{T}\int_0^T [x^T \quad u_2^T]\begin{bmatrix} C_1^T \\ D_{12}^T \end{bmatrix}[C_1 \quad D_{12}]\begin{bmatrix} x \\ u_2 \end{bmatrix}dt\right\} \end{aligned}$$

对应的白噪声协方差矩阵为：

$$\begin{aligned} E\{\begin{bmatrix} \xi(t) \\ \theta(r) \end{bmatrix}[\xi(t)\,\theta(r)]^T\} &= \begin{bmatrix} \Xi & N_f \\ N_f{}^T & \Theta \end{bmatrix} = \begin{bmatrix} B_1^T \\ D_{21} \end{bmatrix}[B_1^T \quad D_{21}^T]\delta(t-\tau) \\ &= \begin{bmatrix} B_1B_1^T & B_1D_{21}^T \\ D_{21}B_1^T & D_{21}D_{21}^T \end{bmatrix}\delta(t-\tau) \end{aligned}$$

求解H_2优化控制问题等价的 LQG 优化问题，根据分离原理，得到如下的 Kalman 滤波器方程:

$$\dot{\hat{x}} = A\hat{x} + B_2 u_2 + K_f(y_2 - C_2\hat{x} - D_{22}u_2)$$

$$K_f = (\Sigma C_2^T + N_f)\Theta^{-1} = (\Sigma C_2^T + B_1 D_{21}^T)(D_{21}D_{21}^T)^{-1}$$

全状态反馈具有下面的形式:

$$u_2 = K_c\hat{x}$$

$$K_c = R^{-1}(B_2^T P + N_c) = (D_{12}^T D_{12})^{-1}(B_2^T P + D_{12}^T C_1)$$

其中，$\Sigma = \Sigma^T$ 和 $P = P^T$ 分别满足如下的代数 Riccati 方程:

$$A^T P + PA - (PB_2 + N_c)R^{-1}(B_2^T P + N_c^T) + Q = 0$$

$$\Sigma A^T + A\Sigma - (\Sigma C_2^T + N_f)\Theta^{-1}(C_2\Sigma + N_f^T) + \Xi = 0$$

2 有关函数说明

函数 h2lqg 和 dh2lqg 分别用于计算连续和离散H_2优化控制器，它们的使用说明如下:

- h2lqg

功能: 连续系统H_2最优控制器综合。

格式: [acp,bcp,ccp,dcp,acl,bcl,ccl,dcl] = h2lqg(A,B1,B2,...D22)

[acp,bcp,ccp,dcp,acl,bcl,ccl,dcl] = h2lqg(A,B1,B2,...D22,aretype)

[sscp,sscl] = h2lqg(TSS)

[sscp,sscl] = h2lqg(TSS,aretype)

说明: 输入参数定义为:

A、B1、B2、C1、C2、D11、D12、D21、D22——双端口连续系统的状态空间矩阵;

TSS——双端口系统定义的树型模型变量;

aretype——用于指定求解代数 Riccati 方程的方法，其取值如下:

aretype='eigen'——特征向量方法 ，缺省值;

aretype='Schur'——Schur 向量方法。

输出参数定义为:

acp,bcp,ccp,dcp——状态反馈传递函数矩阵的状态空间实现;

acl,bcl,ccl,dcl——闭环系统的状态空间矩阵;

sscp1——状态反馈矩阵对应的状态空间模型变量;

sscc1——闭环系统的状态空间模型变量。

- dh2lqg

功能: 离散系统H_2最优控制器综合。

格式： [acp,bcp,ccp,dcp,acl,bcl,ccl,dcl] = dh2lqg(A,B1,B2,...D22)

[acp,bcp,ccp,dcp,acl,bcl,ccl,dcl] = dh2lqg(A,B1,B2,...D22,aretype)

[sscp,sscl] = dh2lqg(TSS)

[sscp,sscl] = dh2lqg(TSS,aretype)

说明： 输入参数与输出参数的定义与 h2lqg 相同。

对于上述两个函数需要加以说明的有以下几点：

1)（A，B2，C2）必须是可镇定的和可测的；

2) 对连续系统的 H_2 优化问题，要求矩阵 D11 必须为 0，如果输入参数中的 D11 不为 0，函数 h2lqg 将自动地把 D11 作为零矩阵处理；

3) D12 和 D21 的转置必须是列满秩的。

3.4.3 连续/离散 H_∞ 综合

1 H_∞ 控制问题描述

对于双端口系统 $P(s)$：

$$P=\begin{bmatrix} A & B_1 & B_2 \\ C_{11} & D_{11} & D_{12} \\ C_{21} & D_{21} & D_{2} \end{bmatrix}$$

计算控制器 $F(s)$ 构成图 3.4.2 所示的闭环控制系统，并满足下面的性能指标：

$$\left\|T_{y1u1}\right\|_\infty = \sup_\omega \bar{\sigma}(T_{y1u1}(j\omega)) < 1$$

为求解上述 H_∞ 控制综合问题，自 20 世纪 80 年代以来提出了多种方法，其中早期的方法将求解过程分为三个主要步骤，即：

1）求出对象的增广模型（参见函数 augtf 和 augss）；

2）Youla 参数化（参见函数 Youla）；

3）基于最优反因果描述 Hankel 逼近的插值。

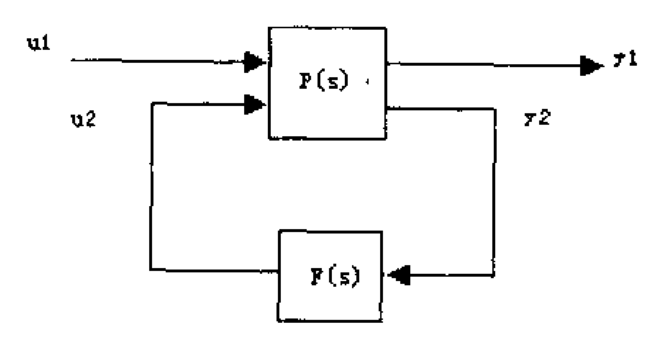


图 3.4.2　H_∞ 闭环控制系统

基于 Youla 参数化和 Hankel 逼近的求解方法存在控制器模型阶次过高的缺点，限制了 H_∞ 控制方法的实用化。为此，有关学者在 1988 年提出了著名的“2-Riccati 方程”方法。该方法只需要求解 2 个非耦合的代数 Riccati 方程，且控制器阶次不超过广义对象的 McMillan 阶次。

求解 H_∞ 控制问题的“2-Riccati 方程”方法可以分别由两条不同的研究途径导出，一条是求解 H_∞ 控制问题的对策理论方法的研究，另一条途径是全通嵌入方法的研究。由上述两条途径导出“2-Riccati 方程”方法都需要经过繁杂的推导和计算，以及前提条件的设定。为解决这一问题，Safonov 于 1989 年提出了一种简化的推导方法，即回路平移（Loop Shifting）技术。该方法消除了前面两种方法的数值不稳定性。

对于离散 H_∞ 控制问题的求解方法，可以采用双线性变换（参见函数 bilin）将离散系统变换为具有等价 H_∞ 范数的连续系统。在对连续系统求解 H_∞ 控制问题得到相应的控制器后，进行逆向的双线性变换即可得到离散 H_∞ 控制问题的解。

鲁棒控制工具箱提供了 3 个函数用于求解连续和离散系统的 H_∞ 控制问题，即 hinf、linf 和 dhinf。其中，hinf 采用基于回路平移（Loop Shifting）技术的“2-Riccati 方程”方法计算连续系统 H_∞ 控制器，linf 采用基于 Youla 参数化和 Hankel 逼近的求解方法计算连续系统 L_∞ 控制器，dhinf 则用于计算离散系统的 H_∞ 控制器。

2 有关函数说明

- hinf

功能：基于回路平移（Loop Shifting）技术的“2-Riccati 方程”方法计算连续系统 H_∞ 控制器。

格式：
```
[acp,...acl,...hinfo,ak,...dk22] = hinf(A,...D22)
[acp,...acl,...hinfo,ak,...dk22] = hinf(A,...D22,au,...du)
[acp,...acl,...hinfo,ak,...dk22] = hinf(A,...D22,au,...du,verbose)
[sscp,sscl,hinfo,tssk] = hinf(TSSP)
[sscp,sscl,hinfo,tssk] =hinf(TSSP,ssu,)
[sscp,sscl,hinfo,tssk] = hinf(TSSP,ssu,verbose)
```

说明：输入参数定义为：

A,...D22——双端口系统的状态空间矩阵；

au,bu,cu,du——用于参数化 H_∞ 控制器的传递函数矩阵的状态空间实现，由于 H_∞ 控制器是不唯一的，函数 hinf 能够根据参数化矩阵 $U(s)$ 计算全解控制器的参数化实现 $K(s)$，其中 $\|U(s)\|_\infty<1$。相应的闭环系统如图 3.4.3 所示；

如果不指定参数 au、bu、cu 和 du，则函数 hinf 令 $U(s)=0$；

参数 verbose 用于指定计算过程中是否显示有关的结果，若 verbose=1,则显示结果；若 verbose=0，则不显示结果。

输出参数的定义为：

acp,bcp,ccp,dcp——一个控制器特解的状态空间实现；

acl,bcl,ccl,dcl——闭环系统的状态空间实现；

ak1,bk1,bk2,ck1,ck2,dk11,dk12,dk21,dk22——全解 H_∞ 控制器的参数化实现；

hinfo——有关计算过程的信息。

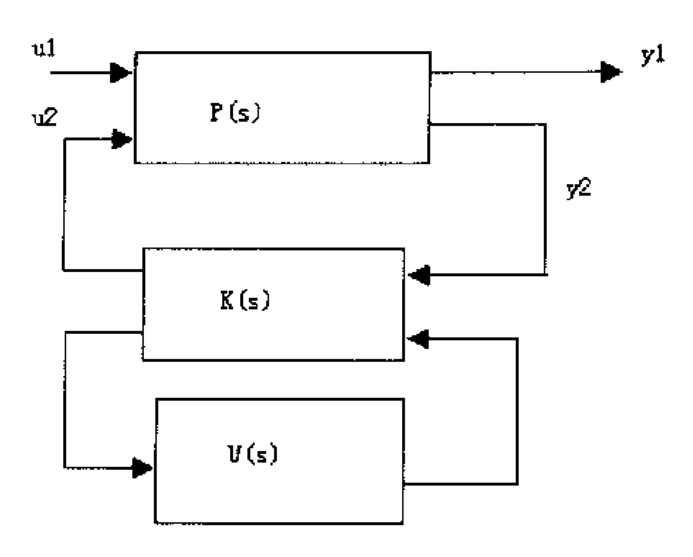


图 3.4.3 具有参数化全解 H_∞ 控制器的闭环系统

- dhinf

功能：求解离散系统的 H_∞ 控制问题。

格式：[acp,...acl,...hinfo,ak,...dk22] = dhinf(A,...D22)

[acp,...acl,...hinfo,ak,...dk22] = dhinf(A,...D22,au,...du)

[acp,...acl,...hinfo,ak,...dk22] = dhinf(A,...D22,au,...du,verbose)

[sscp,sscl,hinfo,tssk] = dhinf(TSSP)

[sscp,sscl,hinfo,tssk] =dhinf(TSSP,ssu,)

[sscp,sscl,hinfo,tssk] =dhinf(TSSP,ssu,verbose)

说明：函数 dhinf 与 hinf 具有相同的参数定义，该函数基于双线性变换计算离散系统的 H_∞ 控制问题。

- linf

功能：基于 Youla 参数化和最优 Hankel 逼近的 L_∞ 控制器综合。

格式：[acp, bcp, ccp, dcp, acl, bcl, ccl, dcl]=linf(A, B1, B2, C1, C2, D11, D12, D21, D22)

说明：输入参数定义为：

A, B1, B2, C1, C2, D11, D12, D21, D22——增广对象的状态空间矩阵。

输出参数定义为：

acp, bcp, ccp, dcp——L_∞ 控制器；

acl, bcl, ccl, dcl——闭环控制系统状态空间矩阵。

3.4.4 H_∞综合的 γ 迭代方法

H_∞ 优化问题是指计算系统的 H_∞ 控制器 $F(s)$且满足如下的性能指标：

$$\min\left\|T_{y1u1}(j\omega)\right\|_\infty$$

鲁棒控制工具箱中的函数 hinfopt 采用基于二分法搜索的 γ迭代方法来计算上述 H_∞ 优化问题。

- hinfopt

功能：基于 γ迭代方法计算 H_∞ 最优控制器。

格式：[gamopt,acp,...,dcp,acl,...,dcl] = hinfopt(A,...,D22)

[gamopt,acp,...,dcp,acl,...,dcl] = hinfopt(A,...,D22,gamind)

[gamopt,acp,...,dcp,acl,...,dcl] =hinfopt(A,...,D22,gamind,aux)

[gamopt,sscp,sscl] = hinfopt(tss)

[gamopt,sscp,sscl] = hinfopt(tss,gamind)

[gamopt,sscp,sscl] = hinfopt(tss,gamind,aux)

说明：输入参数定义为：

A,B1,B2,C1,C2,D11,D12,D21,D22——增广对象的状态空间矩阵；

gamind——用于指定被 γ尺度化的输出通道，缺省值为[1:n]；

aux——用于指定 γ迭代方法的停止条件和 γ迭代的 γ值范围，其格式为：

aux=[tol maxgam mingam]

其中，tol 指定 γ迭代的停止条件，即当相邻两次的稳定解的 γ相对误差小于 tol 时，则停止迭代过程；maxgam 和 mingam 分别用于指定 γ迭代的最大 γ值和最小 γ值；

tss——增广对象的状态空间模型变量。

输出参数定义为：

gamopt——迭代得到的最优 γ值；

acp.bcp.ccp.dcp——控制器的状态空间矩阵；

acl,bcl,ccl,dcl——闭环系统的状态空间矩阵；

sscp＝mksys(acp,bcp,ccp,dcp,'ss')

sscl=mksys(acl,bcl,ccl,dcl,'ss')

举例：考虑如下的 SISO 对象：

$$G(s)=\frac{s-1}{s-2}$$

采用混合灵敏度指标，对应的灵敏度加权矩阵分别为 W_1、W_2 和 W_3（参见 augtf)。

下面对 W_1、W_2 和 W_3 的三种不同情形分别计算 H_∞ 最优控制器。

1） $W_1=\dfrac{0.1(s+1000)}{100s+1}$，$W_2$=0.1，无 W_3：

```
[ag,bg,cg,dg] = tf2ss([1 -1],[1 -2]);
ssg = mksys(ag,bg,cg,dg);
w1 = [0.1*[1 100];[100 1]]; w2 = [0.1;1]; w3 = [];
```

```
[TSS] = augtf(ssg,w1,w2,w3);
[gamopt,sscp,sscl] = hinfopt(TSS,[1:2],[0.001,1,0]);
```

经过 γ迭代计算得到的最优 γ值为 1.5146。

2) 仅有 $W_2=0.1$：

```
w1 = [];
[TSS] = augtf(ssg,w1,w2,w3);
[gamopt,sscp,sscl] = hinfopt(TSS,1,[0.001,1,0]);
```

计算得到的最优 γ值为 2.5,控制器为 F(s)=-4/3。

3) 仅有 $W_1=\dfrac{s+1}{10s+1}$

采用下面的 Matlab 语句：

```
w1 = [1 1;10 1]; w2 = [];w3=[ ];
[TSS] = augtf(ssg,w1,w2,w3);
[gamopt,sscp,sscl] = hinfopt(TSS,1,[0.001,1,0]);
```

计算得到的最优 γ值为 11/6。

3.4.5 LQG 回路传输恢复

1 问题描述

在 LQG 最优控制器设计中，求解两个独立的 Riccati 方程就可得到带有 Kalman 滤波器的 LQG 控制器。但这样设计出来的控制器鲁棒性往往较差，若模型存在微小的偏差或微小的扰动，闭环系统就可能出现不稳定现象。解决上述问题的一个有效方法就是引入回路传输恢复（Loop Transfer Recovery: LTR）技术，其设计思想是设计滤波器增益(或控制器增益)，使得 LQG 闭环控制系统控制量或输出的特性（返回比）尽量逼近对应的 LQR 闭环系统特性（返回比）。

实现全状态反馈回路传输恢复（即控制量返回比逼近）的 LQG/LTR 控制器设计的步骤主要有：

(1) 对给定的开环对象进行 LQR 控制器设计，并调整加权矩阵 Q 和 R 使得返回比矩阵：$L(s)=K_c(sI-A)^{-1}B$ 满足设计要求；

(2) 假设 LQG 问题的白噪声协方差矩阵为：$W=W_0+rI$，其中 W_0 为实际噪声的协方差矩阵，r 为恢复增益；不断增加 r 的取值，设计对应的 LQG 最优滤波器增益 K_f，使得闭环系统的控制量返回比

$$L_r(s)=F(s)G(s)=[K_c(Is-A+BK_c+K_fC-K_fDK_c)^{-1}K_f]G(s)$$

尽量逼近

$$L(s)=K_c(sI-A)^{-1}B$$

实现闭环系统观测器回路传输恢复（输出量返回比逼近）的 LQR/LTR 控制器设计是上述问题的对偶问题，即给定最优滤波器增益 K_f，设计全状态反馈增益矩阵 K_c，使得闭环系统输出量的返回比尽量逼近标准观测器闭环传递函数

$$L_q = C(sI-A)^{-1}K_f$$

2 函数说明

鲁棒控制工具箱的 ltru 和 ltry 分别用于基于控制量和输出量的 LQG 回路传输恢复控制器设计，下面是这两个函数的说明。

- ltru

功能：基于控制量的全状态回路传输恢复。

格式：[af,bf,cf,df,svl] = ltru(A,B,C,D,Kc,Xi,Th,r,w)

[ssf,svl] = ltru(ss,Kc,Xi,Th,r,w,svk)

说明：输入参数定义如下：

A,B,C,D——开环系统状态空间矩阵；

Kc——LQR 全状态反馈增益矩阵；

Xi——实际对象的噪声协方差矩阵；

Th——观测噪声协方差矩阵；

r——恢复增益向量；

w——波特图的频率向量；

svk——全状态反馈闭环传递函数 $L(s)=K_c(sI-A)^{-1}B$ 的频率响应。

输出参数定义如下：

af,bf,cf,df——闭环系统控制器状态空间实现，对应的传递函数为

$$F(s)=[K_c(Is-A+BK_c+K_fC-K_fDK_c)^{-1}K_f]$$

其中，K_f，K_c 分别为滤波器增益矩阵和全状态反馈增益矩阵；

svl——闭环系统的奇异值。

- ltry

功能：基于输出量的观测器回路传输恢复。

格式：[af,bf,cf,df,svl] = ltry(A,B,C,D,Kf,Q,R,q,w)

[ssf,svl] = ltry(ss,Kf,Q,R,q,w,svk)

说明：输入参数定义如下：

A,B,C,D——开环系统状态空间矩阵；

Kf——滤波器增益矩阵；

Q——LQG 性能指标加权矩阵；

R——LQG 性能指标加权矩阵；

q——恢复增益向量；

w——波特图频率向量；

svk——观测器闭环传递函数的频率响应。

输出参数定义如下：

af,bf,cf,df——闭环系统控制器状态空间实现，对应的传递函数为

$$F(s)=[K_c(Is-A+BK_c+K_fC-K_fDK_c)^{-1}K_f]$$

其中，K_f，K_c 分别为滤波器增益矩阵和全状态反馈增益矩阵；

svl——闭环系统的奇异值。

3.4.6 μ 综合

1 μ 综合问题

结构奇异值（μ）方法是鲁棒控制系统分析与综合的有效手段。该方法克服了基于小增益定理的奇异值方法对于未建模动态系统分析和设计的保守性，并能够同时兼顾系统的稳定性和动态性能。

鲁棒控制工具箱的 musyn 函数提供了对多变量系统进行 μ 综合的功能，该函数采用了 D-F 迭代算法来计算控制器 $F(s)$和对角化尺度矩阵 $D(s)$，并满足下面的优化指标：

$$\left\| DT_{y_1u_1}D^{-1} \right\|_\infty < 1$$

D-F 迭代算法的步骤如下：

（1）给定初始对角尺度化矩阵 $D=I$；

（2）采用 H_∞ 优化方法（调用 hinfopt 函数）计算对应的 F（s）；

（3）固定 $F(s)$，采用对角尺度化方法计算闭环系统的结构奇异值，并使得到的对角尺度化矩阵 $D(s)$满足优化指标

$$\mu=\min_{D(j\omega)}\bar{\sigma}(D(j\omega)T_{y_1u_1}D^{-1}(j\omega))$$

(4) 判断 μ 是否小于 1，如果小于 1 则停止计算；

(5) 调用函数 fitd 和 augd 得到新的增广对象，返回 2。

2 函数说明

函数 musyn 的使用说明如下。

- **musyn**

功能：基于 D-F 迭代的 μ 综合。

格式：

```
[acp,...,dcp,mu,logd,ad,...,dd,gam] = musyn(A,B1,B2,...,D22,w)
[acp,...,dcp,mu,logd,ad,...,dd,gam] =
musyn(A,B1,B2,...,D22,w,gammaind,aux,logd0,n,blk, sz,flag)
[sscp,mu,logd,ssd,gam] = musyn(tss,w)
```

[sscp,mu,logd,ssd,gam] = musyn(tss,w,gammaind,aux,logd0,n,blksz,flag)

说明：输入参数定义为：

A,B1,B2,C1,C2,D11,D12,D21,D22——开环双端口对象；

W——计算结构奇异值的频率向量；

gammaind,aux——用于指定 H_∞ 优化的参数（参见 hinfopt）；

logd——用于指定 $D(s)$ 的初始值；

n,blksz,flag——用于指定函数 fitd 的参数（参见 fitd）。

输出参数定义为：

acp,bcp,ccp,dcp——控制器的状态空间实现；

mu——最优的 μ 值；

ad,bd,cd,dd——$D(s)$的状态空间实现；

logd——$D(s)$的对数幅值频率特性；

gam——H_∞ 优化的最终 γ值。

举例：

```
a=2; b1=[0.1,-1]; b2=-1;
c1=[1;.01];
d11=[.1,.2;.01,.01];
d12-[1; 0]; c2=1;
d21=[0,1]; d22=3;
tss=mksys(a,b1,b2,c1,c2,d11,d12,d21,d22,'tss');
w = logspace(-2,1); % FREQUENCY VECTOR
% 开始 μ 综合迭代
[sscp,mu,logd0] = musyn(tss,w);
%显示最优结构奇异值，如图3.4.4所示。
loglog(w,mu);
```

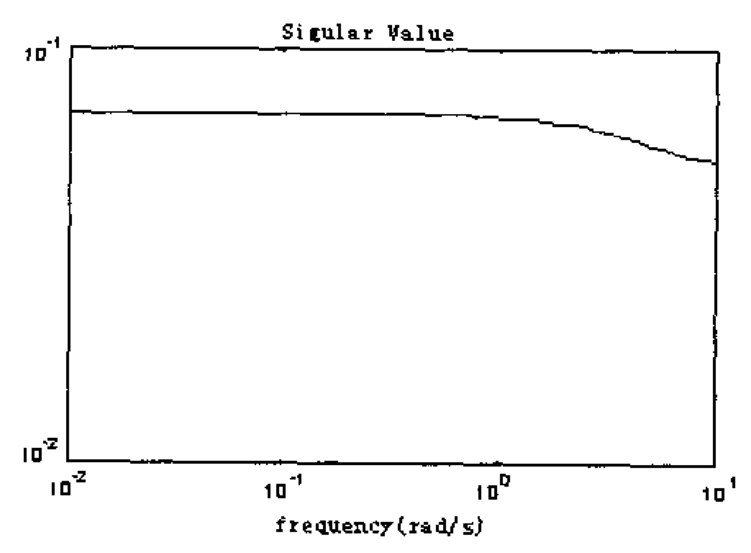


图 3.4.4 最优结构奇异值曲线 1

```
% 采用频率相关的D(s)来改善设计结果
[sscp,mu1,logd1] = musyn(tss,w,[ ],[ ],logd0,1);
% 显示最优结构奇异值，如图3.4.5所示。
loglog(w,mu1);
```

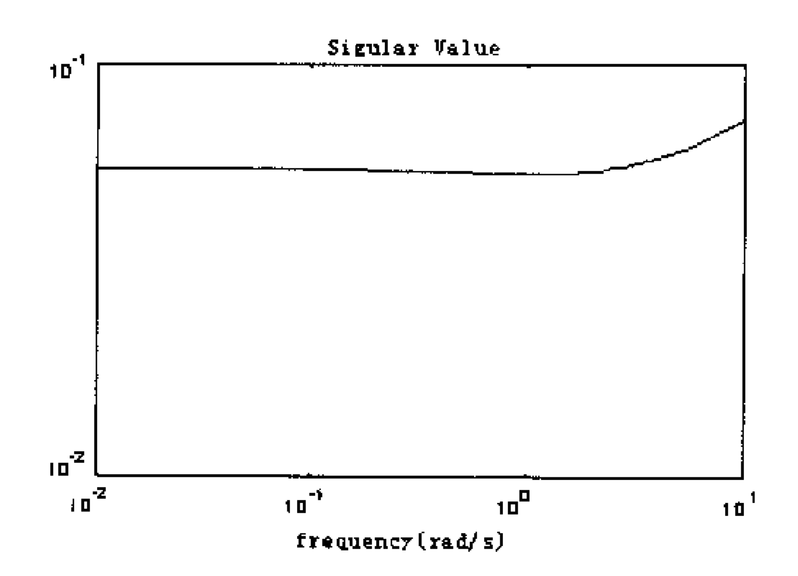


图 3.4.5 最优结构奇异值曲线 2

3.4.7 Youla 参数化

1 问题描述

所谓 Youla 参数化，是指计算所有能够镇定图 3.4.6 所示系统的控制器 $K(s)$。

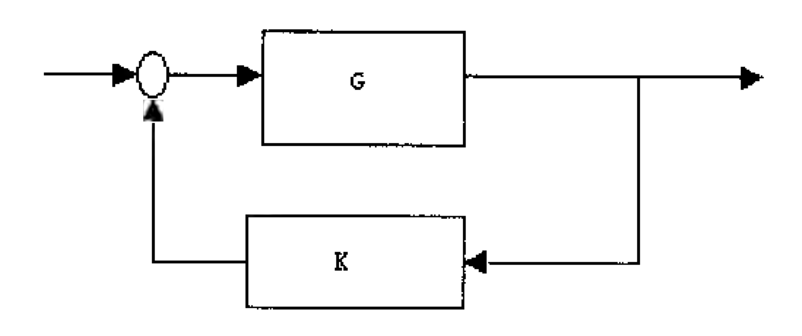


图 3.4.6 Youla 参数化

基于 Hankel 最优逼近的 H_∞ 控制问题求解方法利用了 Youla 参数化，该方法采用了如图 3.4.7 所示的 Youla 参数化即 Q-参数化形式。其中，

$$T_{12}^{T}(-s)T_{12}(s)=I, \quad T_{21}(s)T_{21}^{T}(-s)=I,$$

闭环传递函数为

$$T_{y1u1}=T_{11}(s)+T_{12}(s)Q(s)T_{21}(s)$$

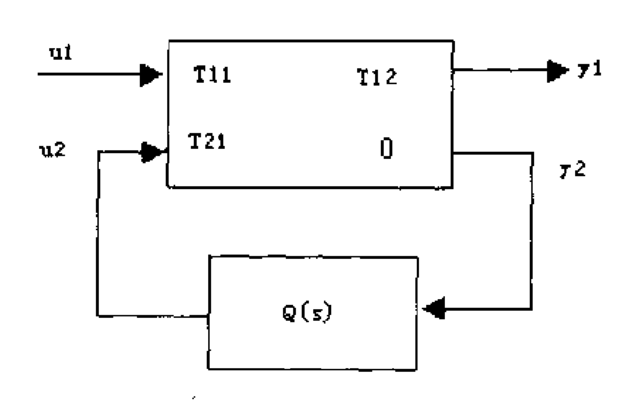


图 3.4.7 Q-参数化

鲁棒控制工具箱的函数 youla 用于计算所有可实现的稳定闭环传递函数矩阵的 Youla 参数化形式，其采用的算法为：

对于给定的开环系统

$$\begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix}$$

计算一个 LQR 控制器 $K(s)$，使得闭环系统传递函数矩阵为

$$\begin{bmatrix} T_{11}(s) & T_{12}(s) \\ T_{21}(s) & T_{22}(s) \end{bmatrix}=\begin{bmatrix} T_{11}(s) & T_{12}(s) \\ T_{21}(s) & 0 \end{bmatrix}$$

其中，

$$T_{12}^{T}(-s)T_{12}(s)=I,\quad T_{21}(s)T_{21}^{T}(\ \ s)=I\ ,$$

同时调用函数 ortc 和 ortr 计算 T_{12} 和 T_{21} 的正交补 $T^{\perp}{}_{12}$ 和 $T^{\perp}{}_{21}$。

2 函数说明

- **youla**

功能： 计算双端口开环系统的所有可实现稳定闭环系统的 Youla 参数化形式。

格式： [at11, bt11, ct11, dt11,at12, bt12, ct12, dt12, at1p,bt1p, ct1p, dt1p,
at21, bt21, ct21, dt21, at2p, bt2p, ct2p, dt2p,kx, x, ky, y, f, h]
=youla(A,B1,B2,C1,C2,D11,D12,D21,D22)

说明： 输入参数定义：

A,B1,B2,C1,C2,D11,D12,D21,D22——双端口系统的状态空间矩阵。

输出参数定义：

at11, bt11, ct11, dt11——传递函数 T_{11} 的状态空间实现；

at12, bt12, ct12, dt12——传递函数T_{12}的状态空间实现；

at1p, bt1p, ct1p, dt1p——传递函数$T_{12}^{\perp}$的状态空间实现；

at21, bt21, ct21, dt21——传递函数T_{21}的状态空间实现；

at2p, bt2p, ct2p, dt2p——传递函数的$T_{21}^{\perp}$状态空间实现；

kx,x,ky,y,f,h——连续LQR综合的计算结果，对应的Matlab语句为：

```
[kx,x] = lqrc(A,B2,C1'*C1,D12'*D12,C1'*D12);
[ky,y] = lqrc(A',C2',B1*B1',D21*D21',B1*D21');
f = -kx;
h = -ky';
```

函数 youla 的计算结果将用于基于 Hankel 最优逼近的H_∞综合。

3.5　模型降阶工具

在复杂系统的控制器设计中，模型降阶往往对于系统设计和成功应用具有重要的意义，主要体现在以下 3 个方面。

（1）在进行控制器设计之前，选择一个有效的系统模型可以大大简化系统设计，模型降阶技术能够简化系统模型，从而进一步简化控制器的设计。

（2）采用某些设计方法得到的控制器往往具有系统的不可观和不可控状态，这些状态必须利用模型降阶方法从控制器中消除。

（3）如果对控制器的复杂性没有显式地加以约束，则设计得到的控制器阶数可能过高而无法实现，只有采用模型降阶方法才能解决这一问题。

目前提出了多种模型降阶方法，如 Shur 降阶方法、最优 Hankel 逼近方法等。Matlab 鲁棒控制工具箱提供了对模型降阶方法的支持，如表 3.5.1 所示。

表 3.5.1　鲁棒控制工具箱的模型降阶函数

函数名称	功　能
balmr	均衡模型降阶
bstschml	Schur 相对误差降阶
imp2ss	脉冲响应向状态空间模型的转换
obalreal	有序均衡实现
ohklmr	最优 Hankel 最小阶逼近降阶
schmr	Schur 模型降阶

3.5.1　均衡（Balanced）模型降阶

系统状态空间模型的均衡实现是指系统的可控和可观 Gram 矩阵具有相同的对角形式，由系统的均衡状态空间实现得到截断的降阶模型具有如下的特性：

$$\|G(s)-G_r(s)\|_\infty \le 2\sum_{i=r+1}^{n}\sigma_i$$

其中，$G(s)$和 $G_r(s)$分别为均衡模型和截断降阶模型，具有下面的形式：

$$G(s)=\begin{bmatrix} A_{11} & A_{12} & B_1 \\ A_{21} & A_{22} & B_2 \\ C_1 & C_2 & D \end{bmatrix} \qquad G_r(s)=\begin{bmatrix} A_{11} & B \\ C_1 & D \end{bmatrix}$$

Gram 矩阵为：$\Sigma=\mathrm{diag}(\sigma_1 I,\ \sigma_2 I,\ \ldots\sigma_m I)$。

由于系统均衡实现的上述性质，因此成为系统模型降阶的一种重要方法。鲁棒控制工具箱的函数 obalreal 用于计算系统状态空间模型的均衡实现。

1 obalreal

功能：系统状态空间模型的有序均衡实现。

格式：[abal,bbal,cbal,g,t] = obalreal(a,b,c)

说明：输入参数定义为：

a,b,c——原系统的状态空间矩阵。

输出参数定义为：

abal,bbal,cbal——系统的有序均衡实现；

g——均衡系统的 Gram 矩阵；

t——均衡实现的状态变换矩阵。

该函数要求原系统是最小实现，否则算法将无法成功执行。

基于均衡模型的降阶方法可以分为基于截断(Truncated)的一般均衡模型降阶和 Schur 均衡模型降阶。这两种方法得到的降阶模型都满足误差的无穷范数界条件，即：

设 n 阶系统 $G(s)=C(sI-A)^{-1}B+D$ 和降阶模型：

$$G_m(s)=C_m(sI-A_m)^{-1}B_m+D_m,$$

$$\|G(j\omega)-G_m(j\omega)\|_\infty \le totbnd$$

$$totbnd=2\sum_{i=k+1}^{n}svh(i)$$

其中，向量 svh 包含了 $G(j\omega)$的稳定和不稳定投影（参见函数 staproj）的 Hankel 奇异值。

函数 balmr 和 schmr 分别用于完成基于截断的模型降阶和 Schur 模型降阶，balmr 计算降阶模型的均衡状态空间实现；函数 schmr 则不计算降阶模型的均衡状态空间实现。

2 balmr

功能：基于截断的均衡模型降阶。

格式：[am,bm,cm,dm,totbnd,svh] = balmr(a,b,c,d,Typ

[am,bm,cm,dm,totbnd,svh] = balmr(a,b,c,d,Type,aug)

[ssm,totbnd,svh] = balmr(ss,Type,aug)

说明：输入参数定义为：

a,b,c,d——系统的状态空间矩阵；

Type——指定参数的类型，aug 为参数：

- Type=1，aug=k——k 为降阶模型的阶数；
- Type=2 , aug=tol——计算一个 k 阶模型，使误差无穷范数上界小于 tol；
- Type=3——显示 Hankel 奇异值向量和阶数 k。

输出参数定义为：

am,bm,cm,dm——降阶模型的均衡状态空间实现；

totbnd——模型误差的无穷范数上界；

svh——$G(j\omega)$的稳定和不稳定投影的 Hankel 奇异值。

3　schmr

功能：Schur 均衡模型降阶。

格式：[am,bm,cm,dm,totbnd,svh] = schmr(a,b,c,d,Type)

[am,bm,cm,dm,totbnd,svh] = schmr(a,b,c,d,Type,aug)

[ssm,totbnd,svh] = schmr(ss,Type,aug)

说明：输入参数定义为：

a,b,c,d——系统的状态空间矩阵；

Type——指定参数的类型，aug 为参数：

- Type=1，aug=k——k 为降阶模型的阶数；
- Type=2 ，aug=tol——计算一个 k 阶模型使误差无穷范数上界小于 tol；
- Type=3——显示 Hankel 奇异值向量和阶数 k。

输出参数定义为：

am,bm,cm,dm——降阶模型的均衡状态空间实现；

totbnd——模型误差的无穷范数上界；

svh——$G(j\omega)$的稳定和不稳定投影的 Hankel 奇异值。

3.5.2　Schur 相对误差模型降阶方法

给定一个 n 阶稳定对象

$$G(s)=\begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

Schur 相对误差模型降阶方法（又称 Schur 均衡随机截断）计算得到的 k 阶降阶模型 $\bar{G}(s)$ 同时满足下面两个误差上界条件，即

乘性误差上界 $\left\|\overline{G}^{-1}(G-\overline{G})\right\|_{\infty} \le 2\sum_{i=k+1}^{n}\frac{\sigma_i}{1-\sigma_i}$

相对误差上界 $\left\|G^{-1}(G-G)\right\|_{\infty} \leqslant 2\sum_{i=k+1}^{n}\frac{\sigma_i}{1-\sigma_i}$

其中，σ_i 为全通相位矩阵 $(W^*(s))^{-1}G(s)$的 Hankel 奇异值，$W(s)$为矩阵$\Phi=G(s)G^T(s)$的左谱因子，即$\Phi=W^T(-s)W(s)$。

函数 bstschmr 和 bstschml 分别用于计算 Schur 相对误差模型降阶及其对偶问题，所谓 Schur 相对误差模型降阶的对偶问题是指具有如下的误差上界条件的模型降阶问题

乘性误差上界 $\left\|(G-\overline{G})\overline{G}^{-1}\right\|_{\infty} \leqslant 2\sum_{i=k+1}^{n}\frac{\sigma_i}{1-\sigma_i}$

相对误差上界 $\left\|(G-\overline{G})G^{-1}\right\|_{\infty} \leqslant 2\sum_{i=k+1}^{n}\frac{\sigma_i}{1-\sigma_i}$

1 bstschmr

功能：Schur 相对误差模型降阶。

格式：
```
[ared,bred,cred,dred,aug,svh] = bstschmr(A,B,C,D,Type)
[ared,bred,cred,dred,aug,svh] = bstschmr(A,B,C,D,Type,no)
[ared,bred,cred,dred,aug,svh] = bstschmr(A,B,C,D,Type,no,info)
[ssred,aug,svh] = bstschmr(SS,Type)
[ssred,aug,svh] = bstschmr(SS,Type,no)
[ssred,aug,svh] = bstschmr(SS,Type,no,info)
```

说明：输入参数定义为：

A，B，C，D——原系统的状态空间模型；

Type——指定参数类型，no 为相应参数，Type 可以取值为：

- Type=1, no=k，k 为降阶模型的阶数；
- Type=2，no=tol，tol 为相对误差的分贝值，使得对于使用频率，下式成立

$$\overline{G}(j\varpi)\subset G(j\omega)\pm tol\ ;$$

- Type=3，显示 Hankel 奇异值向量 svh 和模型阶数 k；

info——指定误差指标的类型，缺省值为 right，即完成 Schur 相对误差模型降阶，当 info=left 时，调用函数 bstschml，完成对偶问题的求解；

ss=mksys(A,B,C,D,'ss')。

输出参数定义为：

ared,bred,cred,dred——降阶模型的状态空间矩阵；

aug——被删除的状态以及相对误差界，其中

- aug(1,1)——被删除的状态；
- aug(1,2)——相对误差界；

svh——Hankel 奇异值向量。

2 bstschml

功能：求解 Schur 相对误差模型降阶的对偶问题。

格式：[ared,bred,cred,dred,aug,svh] = bstschml(A,B,C,D,Type)

[ared,bred,cred,dred,aug,svh] = bstschml (A,B,C,D,Type,no)

[ared,bred,cred,dred,aug,svh] =bstschml (A,B,C,D,Type,no,info)

[ssred,aug,svh] = bstschml (SS,Type)

[ssred,aug,svh] = bstschml (SS,Type,no)

[ssred,aug,svh] = bstschml (SS,Type,no,info)

说明：参数定义与函数 bstschmr 基本相同，只是输入参数 info 取值只能为 left。当函数 bstschmr 的输入参数 info 为 left 时，其功能与函数 bstschml 完全相同。

3.5.3 最优 Hankel 最小阶逼近降阶

基于最优 Hankel 最小阶逼近的模型降阶方法没有采用模型均衡技术，而采用了描述系统实现方式，从而克服了均衡实现可能导致的数值病态问题。下面是对基于最优 Hankel 最小阶逼近的模型降阶方法的问题描述。

给定 n 阶稳定系统

$$G(s) = C(sI - A)^{-1}B + D$$

计算 k 阶稳定的最优 Hankel 最小阶逼近模型

$$G_x(s) = C_x(sI - A_x)^{-1}B_x + D_x$$

并满足

$$\|G - G_x\|_\infty \le totbnd$$

$$totbnd = 2\sum_{i=k+1}^{n}\sigma_i$$

其中，$\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_n$ 为 $G(s)$的 Hankel 奇异值，即 PQ 特征值的平方根，P、Q 分别为(A,B,C,D)的可控和可观 Gram 矩阵。

鲁棒控制工具箱的函数 ohkapp 和 ohklmr 用于计算上述基于最优 Hankel 最小阶逼近的模型降阶问题，其中函数 ohkapp 仅用于计算稳定对象的最优 Hankel 逼近模型；函数 ohklmr 通过调用函数 staproj 和 ohkapp 完成不稳定系统的 k 阶最优 Hankel 最小阶逼近模型的计算。

1 ohkapp

功能：计算稳定对象的最优 Hankel 最小阶逼近降阶模型。

格式：[ax,bx,cx,dx,ay,by,cy,dy,aug] = ohkapp(a,b,c,d,Type)

[ax,bx,cx,dx,ay,by,cy,dy,aug] =ohkapp(a,b,c,d,Type,in)

[ssx,ssy,aug] = ohkapp(ss,Type)

[ssx,ssy,aug] = ohkapp(ss,Type,in)

说明：输入参数定义如下：

a,b,c,d——稳定对象的状态空间矩阵；

Type——指定参数类型：

- Type=1，in=k——k 为降阶模型的阶数；
- Type=2，in=tol——计算一个 k 阶模型使误差无穷范数上界小于 tol；
- Type=3——显示 Hankel 奇异值向量和阶数 k。

输出参数定义为：

ax,bx,cx,dx——降阶模型的状态空间实现；

ay,by,cy,dy——反因果系统 $G_y(s)$的状态空间实现，并满足

$$\left\|G-G_x-G_y\right\|_\infty \le \sigma_{k+1};$$

参数 aug 的各个分量定义为

- aug(1,1)= σ_1；
- aug(1,2)=删除的状态数；
- aug(1,3)=totbnd；
- aug(4:4+n-1)=[$\sigma_1,\sigma_2,\ldots\sigma_n$]。

2 ohklmr

功能：计算不稳定系统的最优 Hankel 最小阶逼近降阶模型。

格式：[am,bm,cm,dm,totbnd,svh] = ohklmr(a,b,c,d,Type)

[am,bm,cm,dm,totbnd,svh] = ohklmr(a,b,c,d,Type,in)

[ssm,totbnd,svh] = ohklmr(ss,..)

说明：输入参数定义如下：

a,b,c,d——稳定对象的状态空间矩阵；

Type——指定参数类型：

- Type=1，in=k——k 为降阶模型的阶数；
- Type=2 , in=tol——计算一个 k 阶模型使误差无穷范数上界小于 tol；
- Type=3——显示 Hankel 奇异值向量和阶数 k。

输出参数定义为：

am,bm,cm,dm——降阶模型的状态空间实现；

totbnd——模型误差的无穷范数上界；

svh——$G(j\omega)$ 的稳定和不稳定投影的 Hankel 奇异值。

3.5.4 脉冲响应向状态空间模型的转换

由系统的脉冲响应序列得到系统的状态空间实现在实际应用中具有重要的意义。鲁棒控制工具箱提供了一种基于 Hankel 奇异值分解的方法来实现上述要求，对应的函数为 imp2ss。

该函数能够根据给定的脉冲响应向量或矩阵生成系统的连续或离散状态空间实现，并保证模型输出误差的无穷范数小于给定的误差容限。该函数采用的算法描述如下。

Step 1　给定系统的脉冲响应序列矩阵（*ny* 维输出，*nu* 维输入，*N*+1 个采样点）：

$$y=[H_0^T(:)\ H_1^T(:)\ldots H_N^T(:)]$$，H_i 为第 i 个采样点的输出向量；

对应的 Hankel 矩阵为

$$\Gamma=\begin{bmatrix} H_1 & H_2 & H_3 & \cdots & H_N \\ H_2 & H_3 & H_4 & \cdots & 0 \\ H_3 & H_4 & H_5 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ H_N & 0 & 0 & \cdots & 0 \end{bmatrix}$$

Step 2　对 Γ 进行奇异值分解得到：

$$\Gamma=U\Sigma V^*=[U_1\ U_2]\begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}\begin{bmatrix} V_1^* \\ V_2^* \end{bmatrix}$$

其中，Σ_1 的维数为 $n\times n$，U_1 和 V_1 的列数均为 n。

将 U_1 和 V_1 进行分块得到下面的结果：

$$U_1=\begin{bmatrix} U_{11} \\ U_{12} \\ U_{13} \end{bmatrix} \qquad V_1=\begin{bmatrix} V_{11} \\ V_{12} \\ V_{13} \end{bmatrix}$$

其中，U_{11}，$U_{13}\in C^{ny\times n}$，V_{11}，$V_{13}\in C^{nu\times n}$；

Step 3　计算原系统的一个离散状态空间实现：

$$A=\Sigma_1^{-\frac{1}{2}}\,\overline{U}\,\Sigma_1^{\frac{1}{2}}$$

$$B=\Sigma_1^{\frac{1}{2}}V_{11}^*$$

$$C=U_{11}\Sigma_1^{\frac{1}{2}}$$

$$D=H_0$$

其中

$$\overline{U}=[U_{11}\ \ U_{12}]\begin{bmatrix} U_{12} \\ U_{13} \end{bmatrix}$$

Step 4　采用双线性变换得到系统的连续状态空间实现：

$$s=\frac{2}{t}\frac{z-1}{z+1}$$

函数 imp2ss 完成上述计算过程，其使用说明如下。

- imp2ss

功能：由脉冲响应计算系统的状态空间实现。

格式：[a,b,c,d,totbnd,svh] = imp2ss(y)

[a,b,c,d,totbnd,svh] = imp2ss(y,ts,nu,ny,tol)

[ss,totbnd,svh] = imp2ss(imp)

[ss,totbnd,svh] = imp2ss(imp,tol)

说明：输入参数定义为：

y——系统的脉冲响应矩阵，N+1 列，ny 行；

ts——采样周期；

nu,ny——输入、输出维数，缺省值 nu=1，ny=size(y)*[1;0]/nu；

tol——模型逼近误差的无穷范数界；

imp——指定输入输出维数的系统脉冲响应模型，由下面的函数构造：

```
imp=mksys(y,t,nu,ny,'imp')
```

输出参数定义为：

a,b,c,d——系统的状态空间实现，当 ts=0 时为离散状态空间实现；当 ts>0 时为连续状态空间实现；

totbnd——模型逼近误差的无穷范数上界，即：

$$\|G-G_N\|_\infty \le totbnd$$

$$totbnd=2\sum_{i=n+1}^{N}\bar{\sigma}_i$$

其中，G，G_N 分别为脉冲响应矩阵的降阶实现和高阶实现；

Svh——Hankel 矩阵 Γ 的奇异值向量。

3.5.5 系统模型降阶的综合应用举例

考虑如下的高阶对象：

$$G(s)=\frac{(s^7+801s^6+1024s^5+599s^4+451s^3+119s^2+49s+5.55)}{s^7+12.6s^6+53.48s^5+90.94s^4+71.83s^3+27.22s^2+4.75s+0.3}$$

采用下面的 Matlab 函数对该对象分别进行均衡模型降阶、Shur 相对误差模型降阶、基于 Hankel 最优逼近的模型降阶，并绘制降阶模型和实际对象的波特图（如图 3.5.1 所示）。在图 3.5.1 中，实线所示为原对象的波特图，虚线为均衡降阶模型的波特图，'*'型曲线为 Hankel 最优逼近降阶模型的波特图。

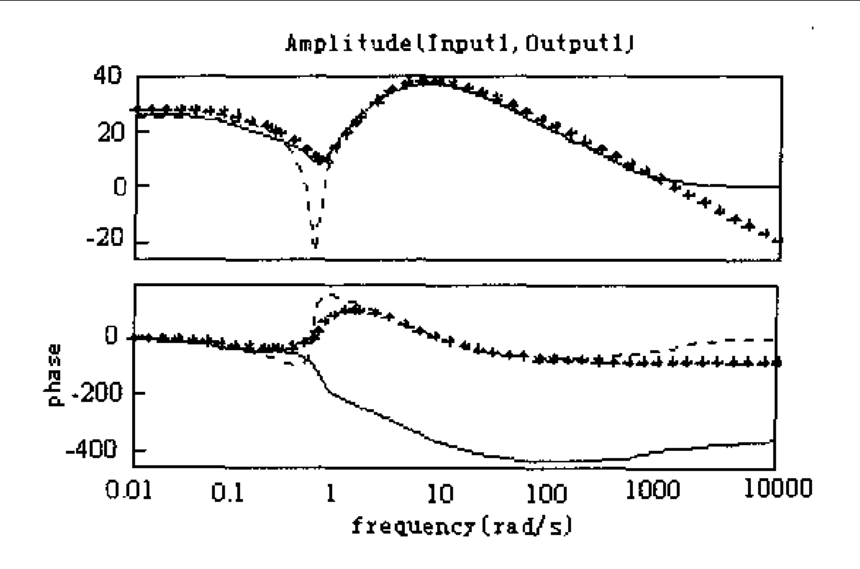


图 3.5.1 原对象和不同降阶模型的波特图

```
num=[1 801  1024 599 451 119 49 5.55]
den=[1 12.6 53.48 90.94  71.83 27.22 4.75 0.3]
[a,b,c,d]=tf2ss(num ,den)
[abal,bbal,cbal,g,t] = obalreal(a,b,c)
Type=3
[am,bm,cm,dm,totbnd,svh] = schmr(a,b,c,d,Type)
[ax,bx,cx,dx,ay,by,cy,dy,aug] = ohkapp(a,b,c,d,Type)
s=ss(a,b,c,d)
sm=ss(am,bm,cm,dm)
sbal=ss(abal,bbal,cbal,d)
sx=ss(ax,bx,cx,dx)
bode(sm,'-',sbal,':',sx,'*')
```

3.6　鲁棒控制工具箱的其他功能函数

在鲁棒控制工具箱中，除了前面 5 节介绍的有关鲁棒控制分析和设计的函数外，还提供了如表 3.6.1 所示的功能函数。

表 3.6.1　鲁棒控制工具箱的其他功能函数

名　称	功　能
aresolv(daresolv)	求解连续（离散）Riccati 方程
riccond (driccond)	求解连续（离散）Riccati 条件数
cshur	计算矩阵的复的 Schur 形式
blkrsch	计算实矩阵的 Schur 形式
iofc（Iofr）	内外因子化
sfl	计算左谱因子
sfr	计算右谱因子

3.6.1 求解连续（离散）Riccati 方程

在鲁棒控制系统分析和设计中，许多问题往往可以归结为代数 Riccati 方程的求解。鲁棒控制工具箱的函数 aresolv 用于求解如下的连续代数 Riccati 方程：

$$A^T P + PA - PRP + Q = 0$$

1 aresolv

功能：求解连续代数 Riccati 方程。

格式：[p1,p2,lamp,perr,wellposed,p] = aresolv(a,q,r)

[p1,p2,lamp,perr,wellposed,p] = aresolv(a,q,r,Type)

说明：输入参数定义为：

a,q,r——代数 Riccati 方程的矩阵 A，Q 和 R；

Type——设定求解方法：

- Type='eigen' ——特征向量方法；
- Type='Schur' ——Schur 向量方法。

输出参数定义为：

p——代数 Riccati 方程的解 P；

p1,p2——矩阵 P_1 和 P_2，满足 $P=P_2P_1^{-1}$；

lamp——闭环特征值；

perr——计算残差；

wellposed——当代数 Riccati 方程对应的 Hamilton 矩阵具有虚轴上的特征值时，取值为 FALSE；否则为 TRUE。

离散代数 Riccati 方程具有下面的形式：

$$A^T PA - P - A^T PB\ (R + B^T PB)^{-1} B^T PA + Q = 0$$

对于离散代数 Riccati 方程的求解，鲁棒控制工具箱提供了函数 daresolv，该函数的使用说明如下。

2 daresolv

功能：求解离散代数 Riccati 方程。

格式：[p1,p2,lamp,perr,wellposed,p] = daresolv(a,b,q,r)

[p1,p2,lamp,perr,wellposed,p] = daresolv(a,b,q,r,Type)

说明：输入输出参数的定义与函数 aresolv 相同。

3.6.2 连续（离散）代数 Riccati 方程的条件数

函数 riccond 和 driccond 分别用于计算连续和离散代数 Riccati 方程的条件数，使用说明

如下。

1　riccond

功能：计算连续代数 Riccati 方程的条件数。

格式：[tot] = riccond(a,b,qrn,p1,p2)

说明：输入参数定义为：

a,b，qrn——代数 Riccati 方程的系数矩阵，对应的 Riccati 方程具有下面的形式：

$$A^T P+PA-(PB+N)R^{-1}(B^T P+N^T)+Q=0$$

p1,p2——[p1;p2]张成 Hamilton 矩阵的稳定特征空间。

输出参数定义为：

tot——与连续代数 Riccati 方程有关的多个条件数构成的向量。

2　driccond

功能：计算离散代数 Riccati 方程的条件数。

格式：[tot] = driccond(a,b,q,r,p1,p2)

说明：输入参数定义为：

a,b,q,r——离散代数 Riccati 方程的系数矩阵，对应的离散代数 Riccati 方程具有如下的形式：

$$A^T PA-P+Q-A^T PB\ (R+B^T PB)^{-1}B^T PA=0$$

p1,p2——[p1;p2]张成 Hamilton 矩阵的稳定特征空间。

输出参数定义为：

tot——与离散代数 Riccati 方程有关的多个条件数构成的向量。

3.6.3　矩阵的 Schur 形式

矩阵的 Schur 形式定义为：对于给定矩阵 A，计算酉矩阵 V，使得 A 的酉相似变换具有形式

$$T=V^T AV=\begin{bmatrix} T_1 & T_{12} \\ 0 & T_2 \end{bmatrix}$$

分块矩阵 T 称为 A 的 Schur 形式。

函数 blkrsch 和 cshur 分别用于计算实矩阵和复矩阵的有序 Schur 形式，它们的使用说明如下。

1　blkrsch

功能：实矩阵的块有序 Schur 形式。

格式：[v,t,m] = blkrsch(a,Type,cut)

说明：输入参数定义为：

a——实的方阵；

Type——指定 Schur 形式的对角块的特征值排序：

- Type=1——$Re(\lambda_i(B_1))<0, Re(\lambda_i(B_2))>0$；
- Type=2——$Re(\lambda_i(B_1))>0, Re(\lambda_i(B_2))<0$；
- Type=3——$Re(\lambda_i(B_1))>Re(\lambda_i(B_2))$；
- Type=4——$Re(\lambda_i(B_1))<Re(\lambda_i(B_2))$；
- Type=5——$|\lambda_i(B_1)|>|\lambda_i(B_2)|$；
- Type=6——$|\lambda_i(B_1)|<|\lambda_i(B_2)|$；

Cut——方阵 B_1 的维数。

输出参数的定义如下：

v——酉矩阵 V；

t——矩阵 A 的 Schur 形式；

m——A 的稳定特征值的数目。

2　cshur

功能：矩阵的有序复 Schur 形式。

格式：[v,t,m,swap] = cschur(a,Type)

说明：输入参数定义为：

a——矩阵 A；

Type——指定复 Schur 形式的对角特征值排序：

- Type=1——$Re(\lambda_i(T_1))<0, Re(\lambda_i(T_2))>0$；
- Type=2——$Re(\lambda_i(T_1))>0, Re(\lambda_i(T_2))<0$；
- Type=3——特征值实部按降序排列；
- Type=4——特征值实部按升序排列；
- Type=5——特征值的模按降序排列；
- Type=6——特征值的模按升序排列。

输出参数定义如下：

v——酉矩阵 V；

t——矩阵 A 的 Schur 形式；

m——A 的稳定特征值的数目。

3.6.4　矩阵的内外因子化（Inner-Outer Factorization）

对于稳定的传递函数矩阵 G，设行维数为 m，列维数为 n，且 $m \ge n$，其内外因子化（Inner-Outer Factorization）形式为

$$G=[\theta\ \ \theta^{\perp}]\begin{bmatrix}M\\0\end{bmatrix}$$

函数 iofr 用于计算矩阵 G（$m \geq n$）的内外因子化。

- iofr

功能：矩阵的内外因子化。

格式：[ain,...,ainp,...,aout,] = iofr(a,b,c,d)

[ssin,ssinp,ssout] = iofr(ss)

说明：输入参数定义为：

a,b,c,d——传递函数矩阵的状态空间实现；

ss——ss=mksys(a,b,c,d,'ss')。

输出参数定义为：

ain,bin,cin,din——θ 的状态空间实现；

apin,bpin,cpin,dpin——$\theta^{\perp}$ 的状态空间实现；

aout,bout,cout,dout——M 的状态空间实现；

ss_in　= mksys(ain,bin,cin,din)；

ss_inp = mksys(ainp,binp,cinp,dinp)；

ss_out = mksys(aout,bout,cout,dout)。

当传递函数矩阵 G 的行维数 m 小于列维数 n 时，函数 iofc 用于计算 G 的内外因子化，采用的算法是首先调用函数 iofr 计算 G 的转置矩阵的内外因子化，然后再变换得到 G 的内外因子化矩阵，其使用格式与 iofr 相同。

3.6.5　计算矩阵的谱因子

传递函数矩阵 G（$\|G\|_{\infty}<1$）的左谱因子 $M(s)$定义为

$$M^{*}(s)M(s)=I-G^{*}(s)G(s)$$

其中，$M(s)$是稳定和最小相位的。

传递函数矩阵 G 的右谱因子定义为

$$M(s)M^{*}(s)=I-G(s)G^{*}(s)$$

函数 sfl 和 sfr 分别用于计算传递函数矩阵 G 的左谱因子和右谱因子。

- sfl（sfr）

功能：计算传递函数矩阵 G 的左（右）谱因子。

格式：[am,bm,cm,dm] = sfl(a,b,c,d)

[am,bm,cm,dm] = sfr(a,b,c,d)

[ssm] = sfl(ss)

[ssm] = sfr(ss);

说明：输入参数定义为：

a,b,c,d——传递函数矩阵的状态空间实现；

ss=mksys(a,b,c,d,'ss')。

输出参数定义为：

am,bm,cm,dm——左(右)谱因子 $M(s)$ 的状态空间矩阵；

ssm=mksys(am,bm,cm,dm,' ss ')。

参 考 文 献

[1] 冯纯伯，田玉平，忻欣，鲁棒控制系统设计，东南大学出版社，1995.12

[2] Ke-min Zhou,John C.Doyle,*Essentials of Robust Control*,Prentice Hall,Inc.,1998

[3] Richard Y. Chiang，Michael G. Safonov， *Matlab Robust Control Toolbox User's Guide* , MathWorks Inc., 1998

第 4 章　模型预测控制工具箱

（Model Predictive Control Toolbox,Ver 1.0.3）

模型预测控制（Model Predictive Control：MPC）是从 20 世纪 80 年代初开始发展起来的一类新型计算机控制算法。该算法直接产生于工业过程控制的实际应用，并在与工业应用的紧密结合中不断完善和成熟。模型预测控制算法由于采用了多步预测、滚动优化和反馈校正等控制策略，因而具有控制效果好、鲁棒性强、对模型精确性要求不高的优点。实际中大量的工业生产过程都具有非线性、不确定性和时变的特点，要建立精确的解析模型十分困难，所以经典控制方法如 PID 控制以及现代控制理论都难以获得良好的控制效果。而模型预测控制具有的优点决定了该方法能够有效地用于复杂工业过程的控制，并且已在石油、化工、冶金、机械等工业部门的过程控制系统中得到了成功的应用。

目前提出的模型预测控制算法主要有基于非参数模型的模型算法控制（MAC）和动态矩阵控制（DMC），以及基于参数模型的广义预测控制（GPC）和广义预测极点配置控制（GPP）等。其中，模型算法控制采用对象的脉冲响应模型，动态矩阵控制采用对象的阶跃响应模型，这两种模型都具有易于获得的优点；广义预测控制和广义预测极点配置控制是预测控制思想与自适应控制的结合，采用对象的 CARIMA（受控自回归积分滑动平均模型），具有参数数目少并能够在线估计的优点，并且广义预测极点配置控制进一步采用极点配置技术，提高了预测控制系统的闭环稳定性和鲁棒性。

Matlab 的模型预测控制工具箱提供了一系列的函数，用于模型预测控制的分析、设计和仿真。这些函数的类型主要有：

(1) 系统模型辨识函数——主要功能包括通过多变量线性回归方法计算 MISO 脉冲响应模型和阶跃响应模型、对测量数据的尺度化等；

(2) 模型建立和转换函数——主要功能包括建立模型预测工具箱使用的系统 mod 模型以及状态空间模型与 mod 模型、阶跃响应模型、脉冲响应模型之间的转换；

(3) 模型预测控制器设计和仿真工具——分为面向阶跃响应模型的预测控制器设计与仿真函数和面向 MPC-mod 模型的设计和仿真函数两类；

(4) 系统分析工具——包括计算模型预测控制系统频率响应、极点和奇异值的有关函数；

(5) 其他功能函数——包括绘图和矩阵计算函数等。

本章将对模型预测工具箱的主要函数的原理和使用方法按以上分类进行介绍，并给出若干设计应用例子。

4.1 系统模型辨识函数

为进行模型预测控制器设计，需要根据系统的输入输出数据建立开环系统的脉冲响应模型或阶跃响应模型，即进行系统模型的辨识。Matlab 模型预测工具箱提供的模型辨识函数如表 4.1.1 所示。

表 4.1.1 系统模型辨识函数

函数名称	功能
autosc	矩阵或向量的自动归一化
imp2step	由 MISO 脉冲响应模型生成 MIMO 阶跃响应模型
mlr	利用多变量线性回归计算 MISO 脉冲响应模型
plsr	利用部分最小二乘方回归方法计算 MISO 脉冲响应模型
rescal	由归一化的数据生成原数据
scal	根据指定的均值和标准差归一化矩阵
validmod	利用新的数据检验 MISO 脉冲响应模型
wrtreg	生成用于线性回归计算的数据矩阵

4.1.1 数据向量或矩阵的归一化

在获得了系统输入输出的原始数据后，为进行参数估计和模型辨识，往往要求对数据进行归一化处理。模型预测工具箱提供的有关数据归一化处理的函数有：autosc、scal 和 rescal，这三个函数都是根据数据的均值和标准差来对数据进行归一化或反归一化处理。设有数据向量：

$$x=[x_1,x_2,\ldots x_n]^T$$

其均值和标准差分别为

$$\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_i,\sigma=\sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n-1}}$$

进行归一化处理后得到的向量为

$$x'=[\frac{x_1-\bar{x}}{\sigma},\frac{x_2-\bar{x}}{\sigma},\ldots\frac{x_n-\bar{x}}{\sigma}]^T$$

1 autosc

功能：矩阵或向量的自动归一化。

格式：[ax,mx,stdx] = autosc(x)

说明：输入参数定义为：

x——数据向量或矩阵。

输出参数定义为：

ax——归一化后的向量或矩阵。若输入参数 x 为矩阵，则对其每一列进行归一化计算；

mx——均值向量。当输入参数 x 为矩阵时，mx 为 x 的各列向量的均值构成的向量；

stdx——标准差向量。当输入参数 x 为矩阵时，stdx 为 x 的各列向量的标准差构成的向量。

举例：
```
x=[1 1 1 2]
[ax,mx,stdx]=autosc(x)
```
输出结果：
```
ax =[-0.5 -0.5 -0.5  1.5]
mx = 1.25
stdx=0.5
```

2 scal

功能：根据指定的均值和标准差进行矩阵或向量的归一化。

格式：sx = scal(x,mx)

sx = scal(x,mx,stdx)

说明：输入参数定义为：

x——数据向量或矩阵；

mx——指定的均值向量。当输入参数 x 为矩阵时，向量 mx 用于指定对 x 的各列向量进行归一化的均值；

stdx——指定的标准差向量。当输入参数 x 为矩阵时，stdx 用于指定对 x 的各列向量进行归一化的标准差。

输出参数定义为：

sx——归一化后的向量或矩阵。若输入参数 x 为矩阵，则对其每一列进行归一化计算。

举例：
```
x=[1 2;2 1]
sx=scal(x, [1 1], [0.5 0.5])
```
输出结果为：
```
sx=[0 2 ;2 0]
```

3 rescal

功能：由归一化的矩阵或向量计算原矩阵或向量（反归一化）。

格式：rx = rescal(x,mx)

rx = rescal(x,mx,stdx)

说明：输入参数定义为：

x——数据向量或矩阵；

mx——用于反归一化的均值向量；

stdx——用于反归一化的标准差向量。

输出参数定义为：

rx——反归一化得到的矩阵或向量。

举例：
```
x=[0 2; 2 0]
rx=rescal(x,[1 1],[0.5,0.5])
```
输出结果：
```
rx=[1 2;2 1]
```

4.1.2 基于线性回归方法的脉冲响应模型辨识

在获得系统的输入输出数据之后，可以采用最小二乘方或部分最小二乘方等线性回归方法来计算系统的脉冲响应模型的系数。在进行线性回归计算之前，需要对原始数据进行预处理，函数 wrtreg 用于完成这一预处理过程。

1 wrtreg

功能：生成用于线性回归计算的输入输出数据矩阵。

格式：[xreg,yreg] = wrtreg(x,y,n)

说明：输入参数定义为：

x——输入数据矩阵；

y——输出数据向量；

n——脉冲响应模型系数的个数。

输出参数定义为：

xreg——预处理后的输入数据矩阵；

yreg——预处理后的输出数据向量。

函数 mlr 和 plsr 分别完成基于多变量最小二乘方和部分最小二乘方方法的脉冲响应模型辨识。

2 mlr

功能：基于多变量最小二乘方的脉冲响应模型辨识。

格式：[theta, yres] = mlr(xreg,yreg,ninput)

[theta, yres] =mlr(xreg,yreg,ninput,plotopt,wtheta,wdeltheta)

说明：输入参数定义为：

xreg——预处理后的输入数据矩阵；

yreg——预处理后的输出数据向量；

ninput——输入变量的个数；

plotopt——绘图选项：

- plotopt=0，缺省值，不绘制图形；
- plotopt=1，绘制实际输出和预测输出；
- plotopt=2，绘制实际输出、预测输出以及输出误差；

wtheta，wdeltheta——最小二乘方的加权向量，缺省值均为 0。

输出参数定义为：

theta——脉冲响应模型的系数矩阵。该矩阵的每一列对应一个输入到输出的脉冲响应模型系数向量；

yres——预测误差向量。

举例：考虑一个双输入单输出的对象，其传递函数矩阵为

$$G(s)=\begin{bmatrix}\dfrac{5.72e^{-14s}}{60s+1}\\ \dfrac{1.52e^{-15s}}{25s+1}\end{bmatrix}$$

采样时间为 7 秒，其输入输出数据存储在数据文件 mlrdat.mat 中；

采用下面的 Matlab 程序对该对象进行脉冲响应模型辨识，系统实际输出和模型的预测输出以及预测误差曲线如图 4.1.1 所示。

```
load mlrdat.mat
[ax,mx,stdx] = autosc(x);
mx = [0 0];
sx = scal(x,mx,stdx);
n = 35;
[xreg,yreg] = wrtreg(sx,y,n);
ninput = 2;
plotopt = 2;
[theta,yres] = mlr(xreg,yreg,ninput,plotopt);
```

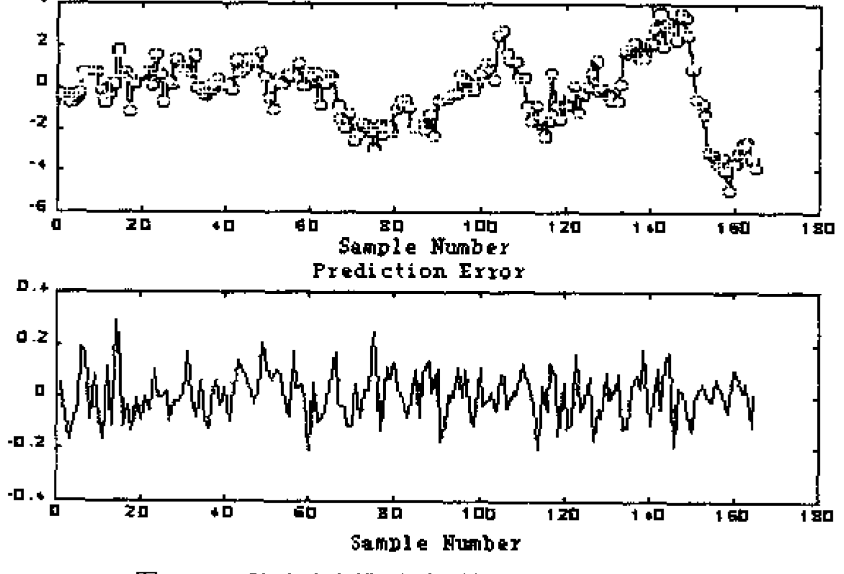


图 4.1.1　脉冲响应模型预测输出与预测误差曲线

3　plsr

功能：基于部分最小二乘方（PLS）方法的脉冲响应模型辨识。

格式：[theta,w,cw,ssqdif,yres] = plsr(xreg,yreg,ninput,lv,plotopt)

说明：输入参数定义为：

xreg——预处理后的输入数据均值；

yreg——预处理后的输出数据向量；

ninput——输入变量的个数；

plotopt——绘图选项：

- plotopt=0，缺省值，不绘制图形；
- plotopt=1，绘制实际输出和预测输出；
- plotopt=2，绘制实际输出；

lv——用于指定最大化输入输出协方差的方向的数目。

输出参数定义为：

w,cw,ssqdif——部分最小二乘方的有关计算结果；

theta——脉冲响应模型的系数矩阵；

yres——预测输出误差。

举例：一个双输入单输出系统的输入输出数据存储在数据文件 plsrdat.mat 中，采用如下的 Matlab 程序对该系统进行模型响应模型辨识，模型的预测输出和预测误差如图 4.1.2 所示。

```
load plsrdat;
n = 30;
[xreg,yreg] = wrtreg(x,y,n);
ninput = 2;
lv = 10;
plotopt = 2;
theta = plsr(xreg,yreg,ninput,lv,plotopt);
```

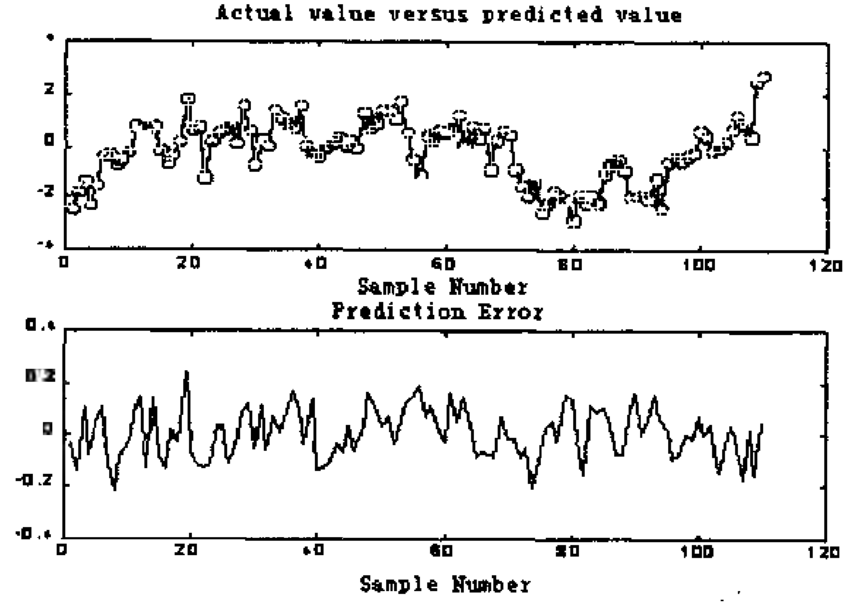


图 4.1.2　基于 PLS 线性回归的模型预测输出与预测误差

4.1.3 脉冲响应模型转换为阶跃响应模型

在模型预测控制工具箱中，模型预测控制器的设计主要基于系统阶跃响应模型。因此在完成系统的脉冲响应模型辨识后，往往需要转换为等价的阶跃响应模型。函数 imp2step 用于完成这一转换过程。

- imp2step

功能： 脉冲响应模型转换为阶跃响应模型。

格式： plant = imp2step(delt,nout,theta1,theta2, ...)

说明： 输入参数定义为：

delt——脉冲响应模型的采样周期；

nout——用于指定输出的稳定性。对于稳定的系统，nout 等于输出的个数；对于具有一个或多个积分输出的系统，nout 为一个长度等于输出个数的向量，该向量对应积分输出的分量为 0，其余分量为 1；

theta1,theta2, ...——脉冲响应系数矩阵，其中 thetai 对应第 i 个输出，i 的最大取值为 25。

输出参数定义为：

plant——系统的阶跃响应系数矩阵，维数为(n×ny+ny+2)×nu，其中，n 为对应每个输入的系数个数，nu 和 ny 分别为输入和输出变量个数。

举例： 考虑下面的双输入单输出的对象：60*s*+1

$$G(s)=\begin{bmatrix}\dfrac{5.72e^{-14s}}{60s+1}\\[2ex]\dfrac{1.52e^{-15s}}{25s+1}\end{bmatrix}$$

采用如下的 Matlab 语句对该系统进行脉冲响应模型辨识，并转换为阶跃响应模型，系统的阶跃响应曲线如图 4.1.3 所示。

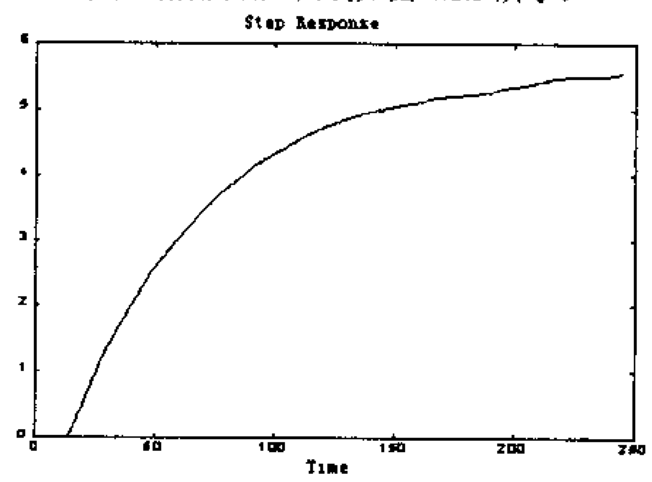


图 4.1.3　系统阶跃响应的预测曲线

```
load mlrdat.mat
[ax,mx,stdx] = autosc(x);
mx = [0 0];
sx = scal(x,mx,stdx);
n = 35;
[xreg,yreg] = wrtreg(sx,y,n);
ninput = 2;
plotopt = 2;
[theta,yres] = mlr(xreg,yreg,ninput,plotopt);
theta = scal(theta,mx,stdx);
nout = 1;
delt = 7;
model = imp2step(delt,nout,theta);
plotstep(model)
```

4.1.4 模型的校验

在根据系统的输入输出数据完成模型辨识以后，可以进一步利用新的数据对辨识模型进行校验，函数 validmod 用于实现这一功能。

- validmod

功能：利用新的数据对辨识模型进行校验。

格式：yres=validmod(xreg,yreg,theta,plotopt)

说明：输入参数定义为：

xreg——经过预处理的输入校验数据；

yreg——经过预处理的输出校验数据；

theta——脉冲响应模型的系数矩阵；

plotopt——绘图选项：

- plotopt=0——缺省值，不绘制图形；
- plotopt=1——绘制实际输出和预测输出；
- plotopt=2——绘制实际输出。

输出参数定义为：

yres——输出预测误差。

4.2 系统模型建立与转换

前面讨论了利用系统输入输出数据进行系统模型辨识的有关函数及使用方法，为进行模型预测控制器的设计，需要对系统模型进行进一步的处理和转换。Matlab 的模型预测工具

箱中提供了一系列函数完成多种模型转换和复杂系统模型的建立功能。在模型预测工具箱中使用了两种专用的系统模型格式，即 MPC mod 模型和 MPC 传递函数模型。这两种模型格式分别是状态空间模型和传递函数模型在模型预测工具箱中的特殊表达形式。这两种模型格式可以同时支持连续和离散系统模型的表达，在 MPC 传递函数模型中还增加了对纯时延的支持。

表 4.2.1 列出了模型预测工具箱的模型建立与转换函数。

表 4.2.1 模型建立与转换函数

函数名称	功能
mod2mod	改变 MPC mod 模型的采样周期
mod2ss	将 MPC mod 模型转换为状态空间模型
mod2step	将 MPC mod 模型转换为阶跃响应模型
poly2tfd	将多项式形式的传递函数模型转换为 MPC 传递函数模型
ss2mod	将状态空间模型转换为 MPC mod 模型
ss2step	将状态空间模型转换为阶跃响应模型
tfd2mod	将 MPC 传递函数模型转换为 MPC mod 模型
tfd2step	将 MPC 传递函数模型转换为阶跃响应模型
th2mod	将 Theta 格式模型（参见辨识工具箱）转换为 MPC mod 模型
addmod	将两个开环模型连接构成闭环模型，使其中一个模型输出叠加到另一个模型的输入
addmd	向对象添加一个或多个测量扰动
addumd	向对象添加一个或多个未测量扰动
appmod	将两个系统模型构成增广系统模型
paramod	将两个系统模型并联
sermod	将两个模型串联

4.2.1 模型建立工具

1 计算闭环系统模型

在模型建立工具中，addmod 用于计算闭环系统的模型。

- **addmod**

功能：计算闭环系统模型。

格式：pmod=addmod(mod1,mod2)

说明：输入参数定义为：

mod1——前向通道的 MPC mod 模型；

mod2——反馈通道的 MPC mod 模型。

输出参数定义为：

pmod ——闭环系统的 MPC mod 模型。

举例：

```
num1=[1 1]
den1=[1 3 1]
```

```
num2=1
den2=[2 1]
[a1,b1,c1,d1]=tf2ss(num1,den1)
[a2,b2,c2,d2]=tf2ss (num2,den2)
mod1=ss2mod (a1,b1,c1,d1)
mod2=ss2mod (a2,b2,c2,d2)
pmod=addmod (mod1,mod2)
```

2 向对象输出添加测量扰动信号

在实际对象的输出端通常叠加有噪声信号。为进行仿真分析，模型预测工具箱的函数 addmd 和 addumd 分别用于向对象的输出添加测量扰动信号模型和未测量扰动信号模型。

- addmd

功能：向对象的输出添加测量扰动信号模型。

格式：model=addmd(pmod,dmod)

说明：输入参数定义为：

pmod——对象的 MPC mod 模型；

dmod——测量扰动的 MPC mod 模型。

输出参数定义为：

model——叠加了噪声的对象 MPC mod 模型。

- addumd

功能：向对象的输出添加未测量扰动信号模型。

格式：model=addumd(pmod,dmod)

说明：输入参数定义为：

pmod——对象的 MPC mod 模型；

dmod——未测量扰动的 MPC mod 模型。

输出参数定义为：

model——叠加了噪声的对象 MPC mod 模型。

3 系统模型的级联和综合

复杂系统的建模往往要求多个系统模型的级联(包括串联、并联)或将两个模型综合为一个整体系统模型，下面对模型预测工具箱的有关函数进行介绍。

- paramod

功能：计算两个系统的并联模型。

格式：pmod=paramod(mod1,mod2)

说明：输入参数定义为：

mod1，mod2——两个并联子系统的 MPC mod 模型。

输出参数定义为：

pmod——并联系统的 MPC mod 模型。

该系统的输出为两个子系统的输出之和，系统的所有输入被重新按类型（包括控制输入、测量扰动和未测量扰动）分组。

- sermod

功能：计算两个系统的串联模型。

格式：pmod=sermod(mod1,mod2)

说明：输入参数定义为：

mod1，mod2——两个串联子系统的 MPC mod 模型。

输出参数定义为：

pmod——串联系统的 MPC mod 模型，在该模型中，系统 mod1 的测量输出作为系统 mod2 的控制输入，两个子系统的扰动信号仍然作为串联系统的扰动信号。

- appmod

功能：计算两个 mod 模型构成的增广系统模型。

格式：pmod=appmod(mod1,mod2)

[pmod,in1,in2,out1,out2]=appmod(mod1,mod2)

说明：输入参数定义为：

mod1，mod2——两个子系统的 MPC mod 模型。

输出参数定义为：

pmod——增广系统的 MPC mod 模型；

in1,in2,out1,out2——增广系统的输入输出变量，其中 in1,out1 为由子系统 mod1 输入输出构成的增广系统输入输出变量；in2,out2 为由子系统 mod2 输入输出构成的增广系统输入输出变量。

4.2.2 模型转换工具

在 Matlab 模型预测工具箱中支持多种系统模型格式，这些模型格式包括：

- 状态空间模型；
- 传递函数模型；
- 阶跃响应模型；
- MPC mod 模型；
- MPC 传递函数模型。

在上述模型格式中，前两种模型格式是 Matlab 通用的模型格式，在其他控制类工具箱中如控制系统工具箱、鲁棒控制工具箱等都予以支持；而后三种模型格式则是模型预测工具箱特有的。其中 MPC mod 模型和 MPC 传递函数模型是通用的状态空间模型和传递函数模型在模型预测工具箱中采用的增广格式，为了方便计算。模型预测工具箱中提供了若干函数，完成上述模型格式之间转换功能。下面对这些函数的用法加以介绍。

1 状态空间模型与 MOC mod 模型之间的转换

MPC mod 模型在状态空间模型的基础上增加了对系统输入输出扰动和采样周期的描述信息，函数 ss2mod 和 mod2ss 实现这两种模型格式之间的转换。

- ss2mod

功能：状态空间模型转换为 MPC mod 模型。

格式：pmod = ss2mod (phi,gam,c,d)

pmod = ss2mod (phi,gam,c,d,minfo)

pmod = ss2mod (phi,gam,c,d,minfo,x0,u0,y0,f0)

说明：输入参数定义为：

phi,gam,c,d——状态空间矩阵；

minfo——构成 MPC mod 模型的其他描述信息，为七个元素的向量，各个元素分别定义为：

- minfo(1) = dt, 系统采样周期，缺省值为 1；
- minfo(2) = n, 系统阶次，缺省值为矩阵 phi 的阶次；
- minfo(3) = nu, 受控输入的个数，缺省值为系统输入的维数；
- minfo(4) = nd, 测量扰动的数目，缺省值为 0；
- minfo(5) = nw, 未测量扰动的数目，缺省值为 0；
- minfo(6) = nym, 测量输出的数目，缺省值为系统输出的维数；
- minfo(7) = nyu, 未测量输出的数目，缺省值为 0。

x0,u0,y0,f0——线性化条件，缺省值均为 0；

如果在输入参数中没有指定 minfo，则取缺省值。

输出参数定义为：

mod——系统的 MPC mod 模型格式。

举例：下面的 Matlab 语句计算具有如下传递函数的系统的 MPC mod 模型格式：

$$G(s)=\frac{s^2+3s+1}{s^3+2s^2+2s^2+1}$$

```
num=[ 1 3 1]
den=[1 2 2 1]
[a,b,c,d]=tf2ss(num,den)
mod=ss2mod(a,b,c,d)
```

输出结果为

```
a =
    -2    -2    -1
    1     0     0
    0     1     0
b =
      1
```

```
      0
      0
c =
     1     3     1
d =
        0
mod=
    1     3     1     0     0     1     0
    NaN   -2    -2    -1    1     0     0
    0     1     0     0     0     0     0
    0     0     1     0     0     0     0
    0     1     3     1     0     0     0
```

- mod2ss

功能：MPC mod 模型转换为状态空间模型。

格式：[phi,gam,c,d] = mod2ss(mod)

[phi,gam,c,d,minfo] = mod2ss(mod)

[phi,gam,c,d,minfo,x0,u0,y0,f0] = mod2ss(mod)

说明：输入参数定义为：

mod——系统的 MPC mod 模型格式。

输出参数定义为：

phi,gam,c,d——状态空间矩阵；

minfo——构成 MPC mod 模型的其他描述信息，其说明参见函数 ss2mod。

举例：将系统 MPC mod 模型重新转换为状态空间模型：

```
[a,b,c,d,minfo,x0,u0,y0,f0]=mod2ss(mod)
```

2　阶跃响应模型与其他模型格式之间的转换

函数 mod2step、tfd2step 和 ss2step 分别用于将 MPC mod 模型、传递函数模型和状态空间模型转换为阶跃响应模型。下面对这三个函数的用法进行说明。

- mod2step

功能：MPC mod 模型转换为阶跃响应模型。

格式：plant = mod2step(mod,tfinal)

[plant, dplant] = mod2step(mod, tfinal, delt2, nout)

说明：输入参数定义为：

mod——系统的 MPC mod 模型；.

tfinal——阶跃响应模型的截断时间；

delt2——采样周期，缺省值由 MPC mod 模型的参数 minfo(1)决定；

nout——输出稳定性向量，用于指定输出的稳定性。对于稳定的系统，nout 等于输出的个数；对于具有一个或多个积分输出的系统，nout 为一个长度等于输出个数的向量，该向量对应积分输出的分量为 0，其余分量为 1。

输出参数定义为：

plant——对象在受控变量作用下的阶跃响应系数矩阵；

dplant——对象在扰动作用下的阶跃响应矩阵。

- **tfd2step**

功能：传递函数模型转换为阶跃响应模型。

格式：plant = tfd2step(tfinal, delt2, nout, g1)

plant = tfd2step(tfinal, delt2, nout, g1, ..., g25)

说明：输入参数定义为：

tfinal——阶跃响应的截断时间；

delt2——采样周期；

nout——输出稳定性向量，参见函数 mod2step 的有关说明；

g1, g2,…——SISO 传递函数，对应多变量系统传递函数矩阵的各个元素按行向量顺序排列构成的向量，其最大个数限制为 25。

输出参数定义为：

plant——对象的阶跃响应系数矩阵。

举例：

例 1　SISO 系统传递函数转换为阶跃响应模型。

设 SISO 系统的传递函数为：

$$G(s)=\frac{s+2}{s^2+3s+1}$$

```
num=[1 2]
den=[1 3 1]
tf1=poly2tfd(num,den,0,0)
plant=tfd2step(5,0.1,1,tf1)
plotstep(plant)
```

由阶跃响应模型绘制的系统阶跃响应曲线如图 4.2.1 所示。

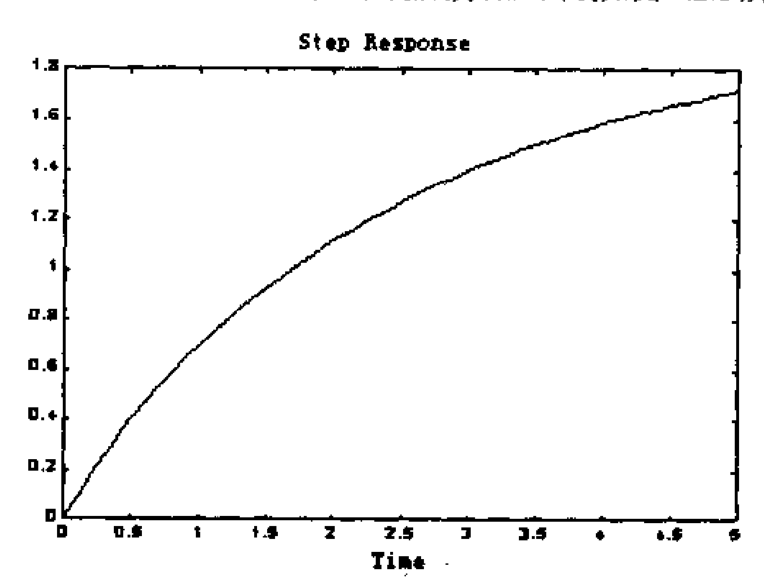


图 4.2.1　系统阶跃响应曲线

例 2　多变量系统传递函数模型转换为阶跃响应模型。

考虑如下的双输入双输出系统的传递函数矩阵：

$$\begin{bmatrix} \dfrac{2}{s+1} & \dfrac{3}{s+2} \\ \dfrac{1}{s+4} & \dfrac{1}{s+1} \end{bmatrix}$$

下面的 Matlab 语句完成传递函数模型向阶跃响应模型的转换：

```
%生成MPC 工具箱的传递函数模型
  tf1=poly2tfd(2,[1,1],0,0)
  tf2=poly2tfd(3,[1 2],0,0)
  tf3=poly2tfd(1,[1,4],0,0)
  tf4=poly2tfd(1,[1,1],0,0)
%转换为阶跃响应模型
  plant=tfd2step(5,0.1,2,tf1,tf2,tf3,tf4)
%绘制系统的阶跃响应曲线
  plotstep(plant)
```

系统在输入 u1 作用下的阶跃响应曲线如图 4.2.2 所示。

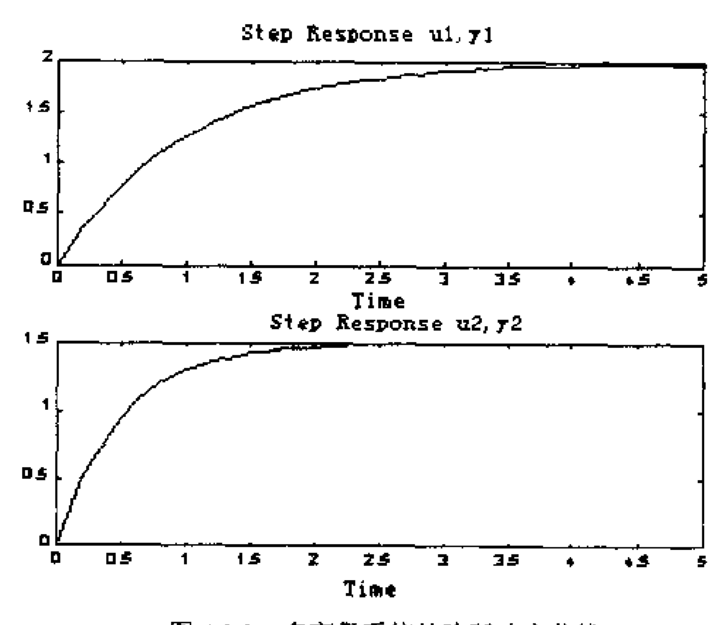


图 4.2.2　多变量系统的阶跃响应曲线

- **ss2step**

功能：状态空间模型转换为阶跃响应模型。

格式：plant = ss2step(phi, gam, c, d, tfinal)

plant = ss2step(phi, gam, c, d, tfinal, delt1, delt2, nout)

说明：输入参数定义为：

phi,gam,c,d——状态空间矩阵；

tfinal——阶跃响应的截断时间；

delt1——系统的采样周期，对于连续系统该参数为 0；

delt2——阶跃响应模型的采样周期，当指定 delt1 时，缺省值为 delt1;当未指定 delt1 时，缺省值为 1；

nout——输出稳定性向量，参见函数 mod2step 的有关说明。

输出参数定义为：

plant——对象的阶跃响应系数矩阵。

举例：考虑如下的系统状态空间模型：

$$A=\begin{bmatrix}-2 & -1\\ 1 & 0\end{bmatrix} \qquad B=\begin{bmatrix}1\\0\end{bmatrix}$$

$$C=\begin{bmatrix}1 & 2\end{bmatrix} \qquad D=0$$

将该模型转换为阶跃响应模型，绘制的阶跃响应曲线如图 4.2.3 所示。

%模型转换，截断时间为 5s,阶跃响应的采样周期为 0.1。

```
plant=ss2step(a,b,c,d,5,0,0.1,1)
```

%绘制阶跃响应曲线

```
plotstep(plant)
```

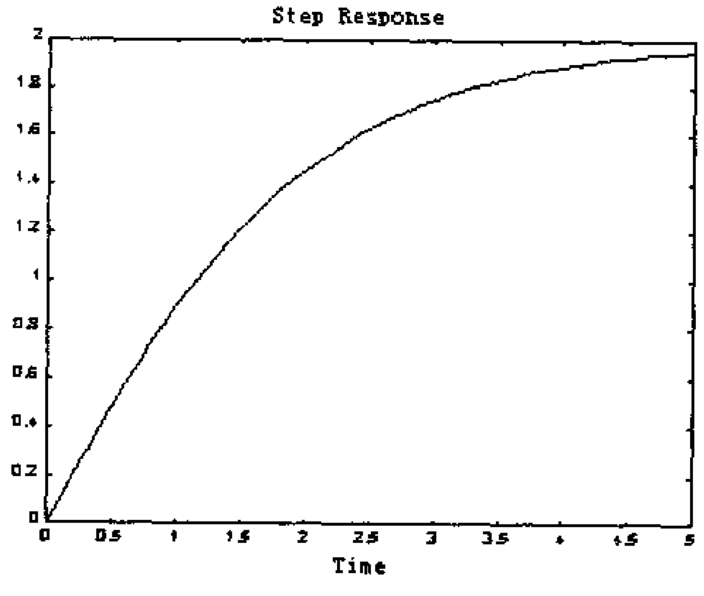


图 4.2.3 系统阶跃响应曲线

3 其他的模型转换函数

在模型预测工具箱中提供的其他模型转换函数还包括通用传递函数模型与 MPC 传递函数模型的转换函数 poly2tfd、MPC 传递函数模型与 MPC mod 模型的转换函数 tfd2mod 和离散 MPC mod 模型之间的转换函数 mod2mod。

- **poly2tfd**

功能：通用传递函数模型与 MPC 传递函数模型的转换。

格式：g = poly2tfd(num,den,delt,delay)

说明：输入参数定义为：

num——传递函数模型的分子多项式系数向量；

den——传递函数模型的分母多项式系数向量；

delt——采样周期，对连续系统，该参数为 0；

delay——系统纯时延，对于离散系统，纯时延为采样周期的整数倍。

输出参数定义为：

g——对象的 MPC 传递函数模型。

举例：考虑如下的纯时延二阶对象：

$$G(s)=\frac{e^{-0.5s}(s+1)}{s^2+4s+4}$$

```
num=[1 1]
den=[1 4 4]
%将多项式形式的传递函数模型转换为 MPC 传递函数格式
g=poly2tfd(num,den,0,0.5)
%将 MPC 传递函数模型转换为阶跃响应模型
plant=tfd2step(5,0.1,1,g)
%绘制系统的阶跃响应曲线
plotstep(plant)
```

系统的阶跃响应曲线如图 4.2.4 所示。

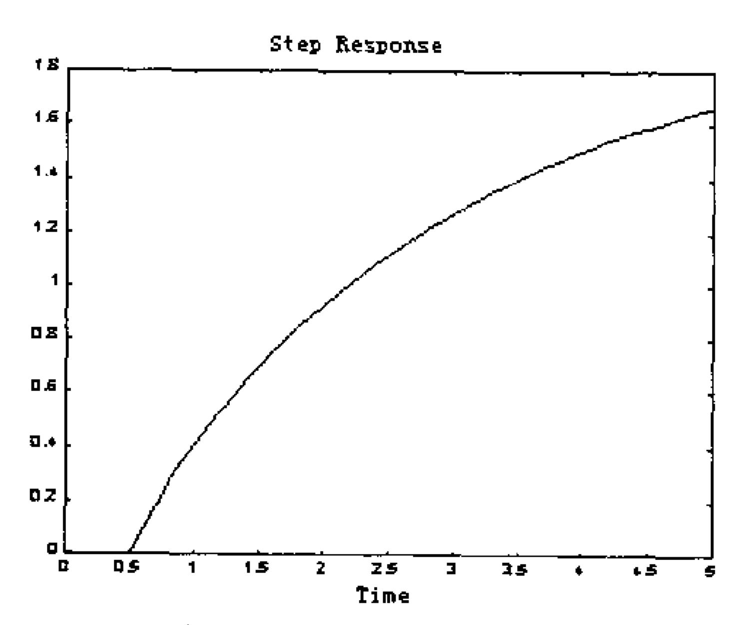


图 4.2.4　纯时延系统的阶跃响应曲线

- **mod2mod**

功能：改变 MPC mod 模型的采样周期。

格式：newmod=mod2mod(oldmod,delt2)

说明：输入参数定义为：

oldmod ——离散系统原有的 MPC mod 模型；

delt2——指定系统新的采样周期。

输出参数定义为：

newmod——改变采样周期后的系统 MPC mod 模型。

举例：考虑二阶对象

$$G(s)=\frac{s+1}{s^2+3s+6}$$

下面的 Matlab 语句用于计算并绘制采样周期为 0.5s 时系统的阶跃响应曲线，如图 4.2.5 所示。

```
num=[1 1]
den=[1 3 6]
% 将多项式模型转换为 MPC 传递函数模型
g=poly2tfd(num,den,0,0)
%将 MPC 传递函数模型转换为连续 MPC mod 模型
mod1=tfd2mod(0.1,1,g)
%改变系统的采样周期
mod2=mod2mod(mod1,0.5)
%计算阶跃响应模型
plant2=mod2step(mod2,5,0.1)
%绘制阶跃响应曲线
plotstep(plant2)
```

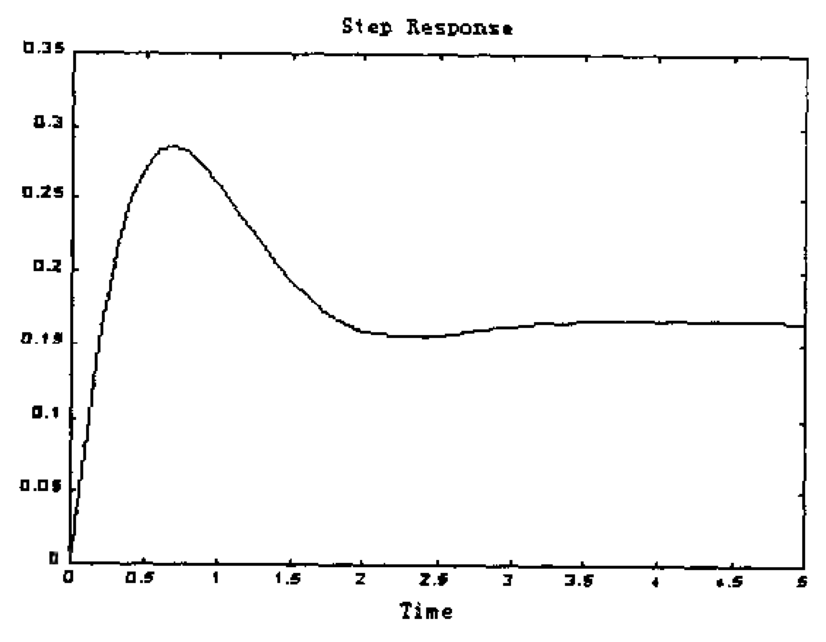


图 4.2.5　二阶对象的阶跃响应曲线

- tfd2mod

功能：MPC 传递函数模型转换为 MPC mod 模型。

格式：pmod = tfd2mod(delt2,ny,g1,g2,g3,…,g25)

说明：输入参数定义为：

delt2——采样周期；

ny——输出个数；

g1, g2,…——SISO 传递函数，对应多变量系统传递函数矩阵的各个元素按行向量顺序排列构成的向量，其最大个数限制为 25。

输出参数定义为：

pmod——系统的 MPC mod 模型。

举例：

```
num=[1 1]
den=[1 3 6]
g=poly2tfd(num,den,0,0)
mod1=tfd2mod(0.1,1,g)
```

- th2mod

功能：系统辨识工具箱的 Theta 格式模型转换为 MPC mod 格式。

格式：umod = th2mod(th)

[umod,emod] = th2mod(th1,th2,th3,th4,th5,th6,th7,th8)

说明：输入参数定义为：

th——系统辨识工具箱的 Theta 格式模型，参见系统辨识工具箱。

输出参数定义为：

umod——系统的测量输入对输出影响的 MPC mod 模型；

emod——系统的噪声输入对输出影响的 MPC mod 模型。

4.3　基于阶跃响应模型的控制器设计与仿真

基于系统的阶跃响应模型进行模型预测控制器设计的方法称为动态矩阵控制方法。该方法的特点是采用工程上易于获取的对象阶跃响应模型，算法较为简单，计算量较少，鲁棒性较强，适用于纯时延、开环渐进稳定的非最小相位系统，在工业部门的过程控制中得到成功的应用。

Matlab 的模型预测工具箱提供了对动态矩阵控制方法的支持，有关的函数能够完成基于阶跃响应模型的模型预测控制器设计和仿真，如表 4.3.1 所示。

表 4.3.1　动态矩阵控制设计与仿真函数

函数名称	功　能
cmpc	输入输出受限的模型预测控制器设计与仿真
mpccon	输入输出不受限的模型预测控制器设计
mpccl	计算模型预测控制系统的闭环模型
mpcsim	模型预测闭环控制系统的仿真（输入输出不受限）
nlcmpc	Simulink 块 nlcmpc 对应的 S 函数
nlmpcsim	Simulink 块 nlmpcsim 对应的 S 函数

4.3.1 输入输出有约束的模型预测控制器设计与仿真

所谓输入输出有约束是指系统的输入输出变量满足一定的上界和下界要求。函数 Cmpc 用于在系统输入输出变量有约束的情况下进行模型预测控制器的设计和仿真。

- cmpc

功能：输入输出受限的模型预测控制器设计与仿真。

格式：[y,u,ym] =

cmpc(plant, model,ywt, uwt,M,P, tend, r, ulim,ylim,tfilter, dplant, dmodel, dstep)

说明：输入参数定义为：

plant——开环对象的实际阶跃响应模型；

model——辨识得到的开环对象阶跃响应模型；

ywt——二次型性能指标的输出误差加权矩阵；

uwt——二次型性能指标的控制量加权矩阵；

M——控制时域长度；

P——预测时域长度，当 P=Inf 时，表示无限的预测和控制时域长度；

tend——仿真的结束时间；

r——输出设定值或参考轨迹。

对应上述参数的二次型性能指标为：

$$J=[Y(k+1)-r(k+1)]^T Q[y(k+1)-r(k+1)]+U^T(k)RU(k)$$

$$Y(k+1)=[y(k+1), y(k+2),..., y(k+P)]^T$$

$$U(k)=[u(k), u(k+1),...,u(k+M-1)]^T$$

其中，Q 为加权矩阵 ywt; R 为加权矩阵 uwt;

（以下的输入参数为可选参数）

ulim——输入控制变量的约束矩阵，包括控制变量的下界、上界和变化率曲线的轨迹；

ylim——输出变量的约束矩阵，包括输出变量的下界和上界的轨迹；

tfilter——噪声滤波器的时间常数和未测扰动的滞后时间常数，缺省值对应无滤波器和阶跃未测扰动的情形；

dplant——输入不可测扰动模型的阶跃响应系数矩阵；

dmodel——输入可测扰动模型的阶跃响应系数矩阵；

dstep——对于输入不可测的扰动，dstep 为扰动模型的输出值；对于可测扰动，dstep 为扰动模型的输入。

输出参数定义为：

y——系统的输出；

u——控制变量；

ym——模型预测输出。

举例：考虑如下的双输入双输出纯时延对象，其传递函数矩阵为：

$$\begin{bmatrix} \dfrac{12.8e^{-1s}}{16.7s+1} & \dfrac{6.6e^{-7s}}{10.9s+1} \\ \dfrac{-19.4e^{-3s}}{14.4s+1} & \dfrac{-18.9e^{-3s}}{21.0s+1} \end{bmatrix}$$

```
%将传递函数模型转换为阶跃响应模型
  g11=poly2tfd(12.8,[16.7 1],0,1);
  g21=poly2tfd(6.6,[10.9 1],0,7);
  g12=poly2tfd(-18.9,[21.0 1],0,3);
  g22=poly2tfd(-19.4,[14.4 1],0,3);
  delt=3;
  ny=2;
  tfinal=90;
  model=tfd2step(tfinal,delt,ny,g11,g21,g12,g22);
%进行模型预测控制器设计
  plant=model;
%预测时域长度为 6
  P=6;
  M=2;
  ywt=[];
  uwt=[1 1];
%设置输入约束和参考轨迹等控制器参数
  r=[0 1];
 %仿真时间为 30
 tend=30;
 [y,u]=mpccon(plant,model,ywt,uwt,M,P,tend,r,ulim,ylim
)
 plotall(y,u,delt)
```

闭环系统的输出曲线和控制变量曲线分别如图 4.3.1 和 4.3.2 所示。

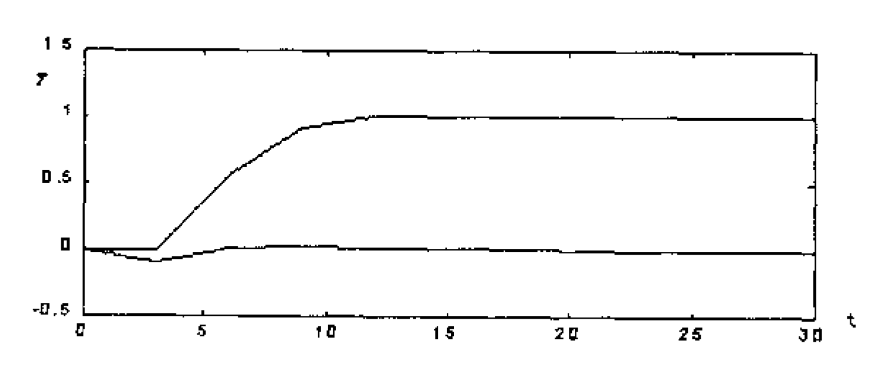


图 4.3.1 闭环系统的输出曲线

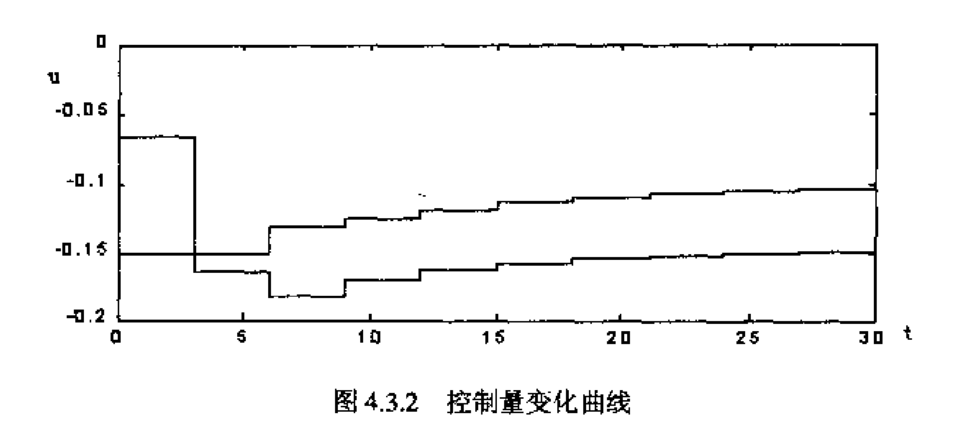


图 4.3.2 控制量变化曲线

4.3.2 输入输出无约束的模型预测控制器设计

前面介绍了系统输入输出有约束条件下的模型预测控制器设计问题。对于输入输出无约束的情况，利用函数 mpccon 可以完成基于系统阶跃响应模型的预测控制器的设计。下面对该函数进行介绍。

1 mpccon

功能：输入输出无约束的模型预测控制器设计。

格式：Kmpc = mpccon(model,ywt,uwt,M,P)

说明：输入参数定义为：

model——开环对象的阶跃响应模型；

ywt——二次型性能指标的输出误差加权矩阵；

uwt——二次型性能指标的控制量加权矩阵；

M——控制时域长度；

P——预测时域长度，当 P=Inf 时，表示无限的预测和控制时域长度。

输出参数定义为：

Kmpc——模型预测控制器的增益矩阵。

举例：考虑具有如下传递函数矩阵的系统：

$$\begin{bmatrix} \dfrac{12.8e^{-1s}}{11.2s+1} & \dfrac{6.6e^{-2s}}{5.2s+1} \\ \dfrac{1.3e^{-5s}}{5s+1} & \dfrac{-1.2e^{-4s}}{3.0s+1} \end{bmatrix}$$

下面的 Matlab 语句对该系统的阶跃响应模型进行输入输出无约束的模型预测控制器设计。

```
g11=poly2tfd(12.8,[11.2 1],0,1);
g21=poly2tfd(6.6,[5.2 1],0,2);
g12=poly2tfd(1.3,[5 1],0,5);
g22=poly2tfd(-1.2,[3.0 1],0,4);
```

```
delt=3;
ny=2;
tfinal=90;
model=tfd2step(tfinal,delt,ny,g11,g21,g12,g22);
%预测时域长度为 6
P=6;
%控制时域长度为 2
M=2;
ywt=[];
uwt=[1 1];
Kmpc=mpccon(model,ywt,uwt,M,P);
r=[0,1];
tend=40;
%进行模型预测控制仿真
[y,u,ym]=mpcsim(model,model,Kmpc,tend,r)
%绘制输入输出变量的变化曲线
plotall(y,u,delt)
```

闭环系统的输出响应曲线和控制量变化曲线分别如图 4.3.3 和 4.3.4 所示。

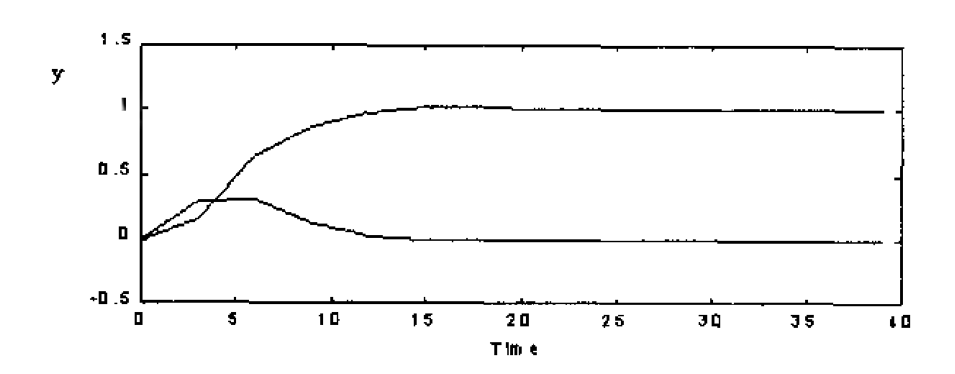


图 4.3.3　闭环系统输出曲线

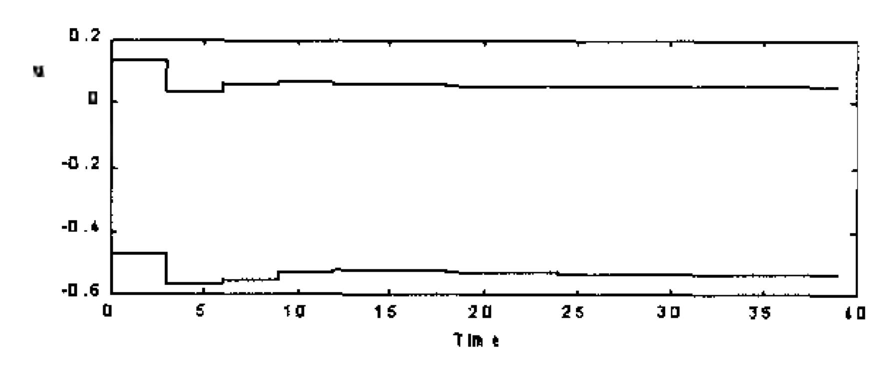


图 4.3.4　控制量变化曲线

在上例中，使用了函数 mpcsim 来进行闭环系统的仿真。该函数的功能就是对无约束的模型预测控制系统进行仿真。

2 mpcsim

功能：输入输出无约束的模型预测控制系统仿真。

格式：[y,u,ym] = mpcsim(plant, model, Kmpc, tend, r,usat, tfilter， dplant, dmodel, dstep)

说明：输入参数定义为：

plant——开环对象的实际阶跃响应模型；

model——辨识得到的开环对象阶跃响应模型；

Kmpc——模型预测控制器的增益矩阵；

tend——仿真的结束时间；

r——输出设定值或参考轨迹；

（以下的输入参数为可选参数）

tfilter——噪声滤波器的时间常数和未测扰动的滞后时间常数，缺省值对应无滤波器和阶跃未测扰动的情形；

dplant——输入不可测扰动模型的阶跃响应系数矩阵；

dmodel——输入可测扰动模型的阶跃响应系数矩阵；

dstep——对于输入不可测的扰动，dstep 为扰动模型的输出值；对于可测扰动，dstep 为扰动模型的输入。

输出参数定义为：

y——系统的输出；

u——控制变量；

ym——模型预测输出。

4.3.3 计算由阶跃响应模型构成的闭环系统模型

当对象和控制器的模型均由阶跃响应形式给定时，函数 mpccl 用于计算闭环系统的 MPC mod 模型。其使用方法说明如下。

- mpccl

功能：计算由阶跃响应模型构成的闭环系统模型。

格式：[clmod,cmod]=mpccl(plant,model,Kmpc,tfilter,dplant,dmodel)

说明：输入参数定义为：

plant——开环对象的实际阶跃响应模型；

model——阶跃响应形式的内部模型；

Kmpc——模型预测控制器的增益矩阵；

（tfilter,dplant,dmodel 为可选参数）

tfilter——滤波器的时间常数和噪声动力学参数构成的矩阵；

dplant——所有扰动（包括可测扰动和不可测扰动）的阶跃响应模型，如果 dplant 为空矩阵，则表示无扰动；

dmodel——可测扰动的阶跃响应模型，如果 dmodel 为空矩阵，则表示无可

测扰动。

输出参数定义为：

clmod——模型预测控制闭环系统的 MPC 状态空间模型（mod 模型）；

cmod——控制器的 MPC mod 模型。

举例：考虑如下的 SISO 系统，其传递函数为：

$$G(s)=\frac{3e^{-4s}}{4s+1}$$

首先绘制开环系统的阶跃响应曲线,如图 4.3.5 所示。

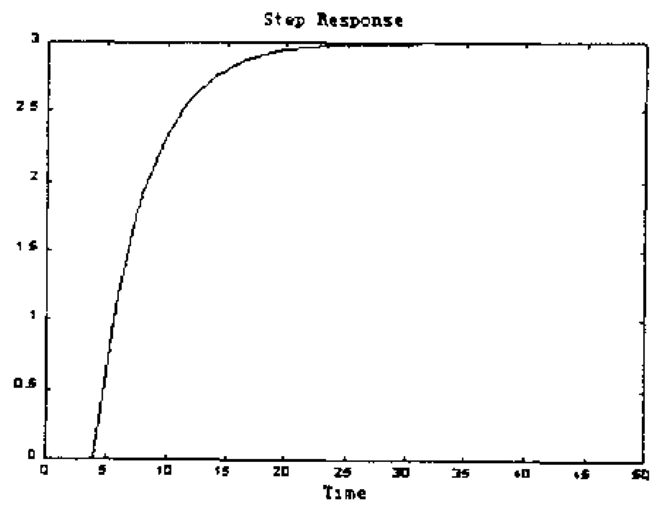


图 4.3.5　开环系统阶跃响应曲线

```
g=poly2tfd(3,[4 1],0,4)
plant=tfd2step(50,2,1,g)
plotstep(plant)
%设计模型预测控制器
%预测时域长度为 6
P=6;
%控制时域长度为 2
M=2;
%设置性能指标的加权阵
ywt=[];
uwt=1;
Kmpc=mpccon(plant,ywt,uwt,M,P);
%计算闭环系统模型
[clmod,cmod]=mpccl(plant,plant,Kmpc)
%绘制闭环系统的阶跃响应
tend=50
%设置参考输入
r=3
[y,u,ym] = mpcsim(plant, plant, Kmpc, tend, r)
```

```
plotall(y,u,2)
```

闭环系统的阶跃响应曲线和控制量变化曲线分别如图 4.3.6 和图 4.3.7 所示。

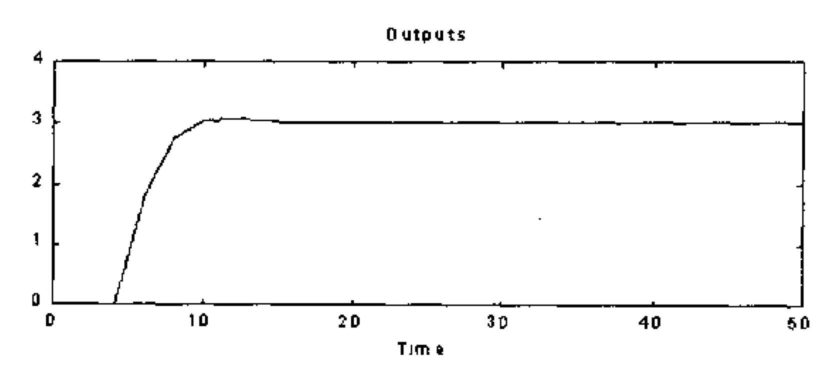


图 4.3.6 闭环系统输出曲线

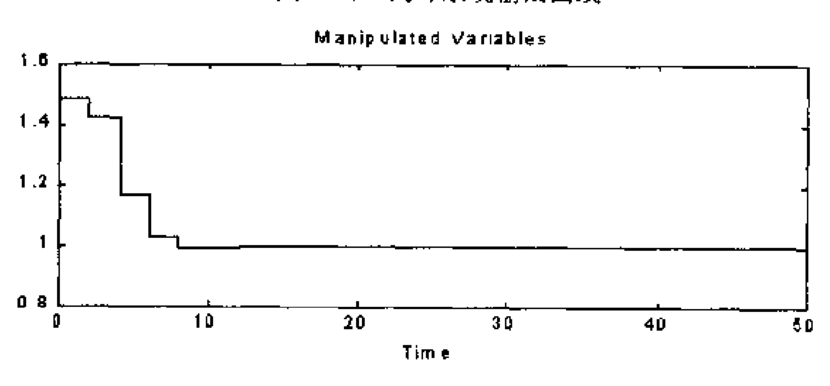


图 4.3.7 控制量变化曲线

4.4 基于状态空间模型的预测控制器设计

在 Matlab 模型预测工具箱中，除了提供基于阶跃响应模型的预测控制器设计功能外，还提供了基于 MPC 状态空间模型（mod 模型）的预测控制器设计功能。有关的函数如表 4.3.1 所示。

表 4.3.1 基于 MPC 状态空间模型的预测控制器设计函数

函数名称	功能
scmpc	输入输出有约束的状态空间模型预测控制器设计
smpccl	计算输入输出无约束的模型预测闭环控制系统模型
smpccon	输入输出无约束的状态空间模型预测控制器设计
smpcest	状态估计器设计
smpcsim	模型预测闭环控制系统仿真

4.4.1 输入输出有约束的状态空间模型预测控制器设计

函数 scmpc 用于进行输入输出有约束条件下的状态空间模型预测控制器设计，该函数的使用说明如下。

- scmpc

  功能：输入输出有约束条件下的状态空间模型预测控制器设计。

格式：[y,u,ym]=scmpc(pmod,imod,ywt,uwt,M,P,tend, r,ulim,ylim,Kest,z,v,w,wu)

说明：输入参数定义为：

pmod——MPC mod 格式的对象状态空间模型，用于仿真；

imod——MPC mod 格式的对象内部模型，用于预测控制器设计；

ywt——二次型性能指标的输出误差加权矩阵；

uwt——二次型性能指标的控制量加权矩阵；

M——控制时域长度；

P——预测时域长度；

ulim ——ulim=[Ulow Uhigh delU]，其中 Ulow 为控制变量的下界，Uhigh 为控制变量的上界，delU 为控制变量的变化率约束；

ylim——ylim=[Ylow Yhigh]，其中 Ylow 为输出的下界，Yhigh 为输出的上界；

Kest——估计器的增益矩阵；

z——测量噪声；

v——测量扰动；

w——输出未测量扰动；

wu——施加到控制输入的未测量扰动。

输出参数定义为：

y——系统响应；

u——控制变量；

ym——模型预测输出。

举例：设系统的传递函数矩阵为

$$\begin{bmatrix} \dfrac{12.8e^{-s}}{16.7s+1} & \dfrac{6.6e^{-7s}}{10.9s+1} \\ \dfrac{-18.9e^{-3s}}{21s+1} & \dfrac{-19.4e^{-3s}}{14.4s+1} \end{bmatrix}$$

```
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
g22=poly2tfd(-19.4,[14.4 1],0,3);
delt=3;
ny=2;
imod=tfd2mod(delt,ny,g11,g21,g12,g22);
pmod=imod;
P=6;
M=2;
ywt=[];
uwt=[1 1];
tend=30;
```

```
r=[0 1];
ulim=[-inf -0.15 inf inf 0.1 100];
ylim=[];
[y,u]=scmpc(pmod,imod,ywt,uwt,M,P,tend,r,ulim,ylim);
plotall(y,u,delt)
```

闭环系统的输出响应曲线和控制量闭环曲线如图 4.4.1 和 4.4.2 所示。

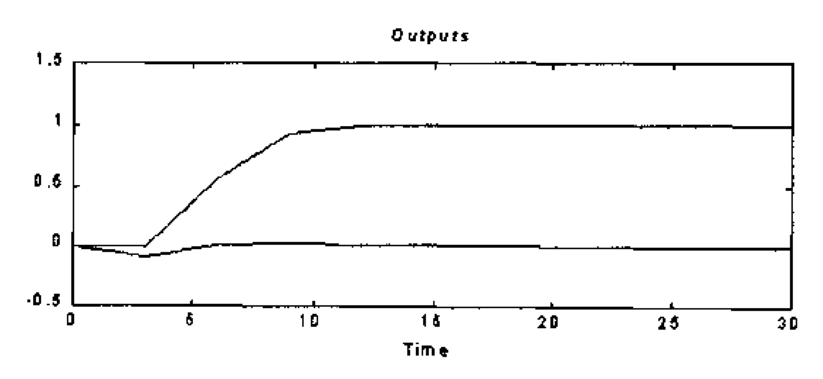


图 4.4.1 闭环系统输出曲线

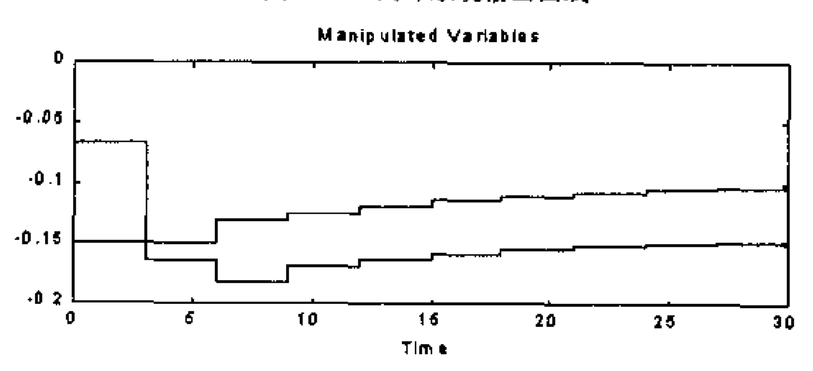


图 4.4.2 控制量变化曲线

在添加对输出变量的约束后，进行模型预测控制器的设计，得到的闭环控制系统输出曲线和控制量变化曲线如图 4.4.3 和 4.4.4 所示。

```
ulim=[-inf -0.15 inf inf 0.1 100];
ylim=[0 0 inf inf];
[y,u]=scmpc(pmod,imod,ywt,uwt,M,P,tend,r,ulim,ylim);
plotall(y,u,delt)
```

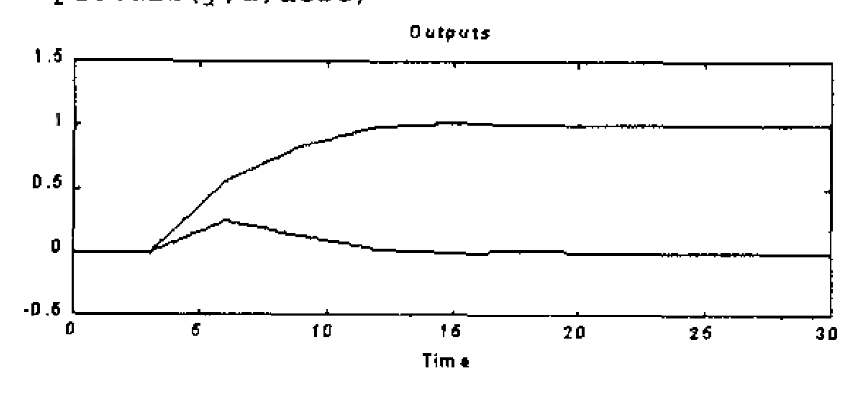


图 4.4.3 输出有约束时的输出响应曲线

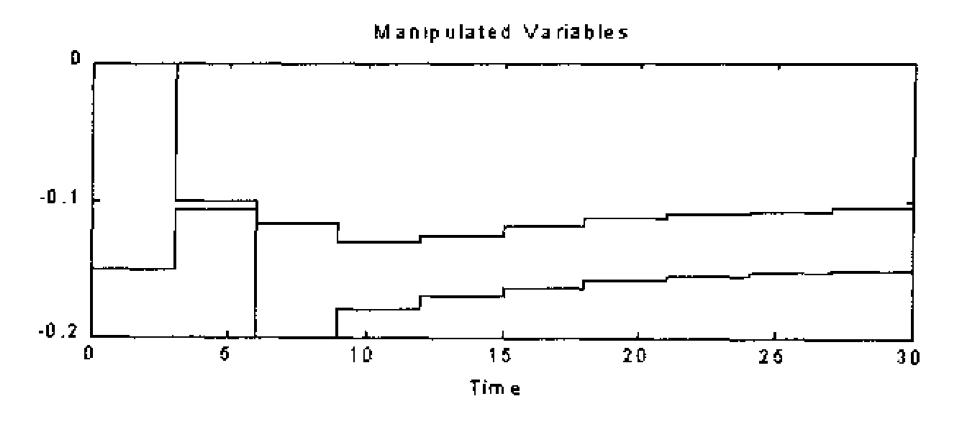


图 4.4.4　控制量变化曲线

4.4.2　输入输出无约束的状态空间模型预测控制器设计

函数 smpccon 用于完成输入输出无约束的状态空间模型预测控制器设计，其输出为预测控制器的增益矩阵。在这一基础上，利用函数 smpcsim 可以对模型预测控制闭环系统进行仿真，在仿真时还可以对控制量施加约束。函数 smpccl 则用于计算闭环系统的 MPC 状态空间模型。

1　smpccon

功能：输入输出无约束的状态空间模型预测控制器设计。

格式：Ks = smpccon(imod,ywt,uwt,M,P)

说明：输入参数定义为：

imod——MPC mod 格式的对象内部模型，用于预测控制器设计；

ywt——二次型性能指标的输出误差加权矩阵；

uwt——二次型性能指标的控制量加权矩阵；

M——控制时域长度；

P——预测时域长度。

输出参数定义为：

Ks——预测控制器的增益矩阵。

2　smpccl

功能：计算输入输出无约束的模型预测闭环控制系统模型。

格式：[clmod,cmod]=smpccl(pmod,imod,Ks)

[clmod,cmod]=smpccl(pmod,imod,Ks,Kest)

说明：输入参数定义为：

pmod——MPC mod 格式的对象状态空间模型；

imod——MPC mod 格式的对象内部模型；

Ks——预测控制器的增益矩阵；

Kest——状态估计器的增益矩阵。

输出参数定义为：

clmod——闭环系统的状态空间模型（MPC mod 格式）；

cmod——预测控制器的状态空间模型（MPC mod 格式）。

3 smpcsim

功能：输入受限的模型预测控制闭环系统设计与仿真。

格式：[y,u,ym]=smpcsim(pmod,imod,Ks,tend,r,ulim,Kest,z,v,w,wu)

说明：输入参数定义为：

pmod——MPC mod 格式的对象状态空间模型，用于仿真；

imod——MPC mod 格式的对象内部模型，用于预测控制器设计；

Ks——预测控制器的增益矩阵；

tend——仿真时间长度；

r——输出设定值；

ulim——输入控制量约束，当 ulim 为 0 或空矩阵时，无输入约束；当对输入施加约束时，ulim=[Ulow Uhigh delU]；

Kest——估计器增益矩阵；

z——测量噪声；

v——可测扰动（或前馈控制）；

w——输出不可测扰动。

wu——输入不可测扰动。

输出参数定义为：

y——系统响应；

u——控制变量；

ym——模型预测输出。

举例：考虑具有如下传递函数矩阵的多变量系统的状态空间模型预测控制器设计问题。

$$\begin{bmatrix} \dfrac{12.8e^{-s}}{16.7s+1} & \dfrac{6.6e^{-7s}}{10.9s+1} \\ \dfrac{-18.9e^{-3s}}{21s+1} & \dfrac{-19.4e^{-3s}}{14.4s+1} \end{bmatrix}$$

```
%在进行模型预测控制器设计之前，首先将系统模型转换为状态空间形式；
T=2;
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
g22=poly2tfd(-19.4,[14.4 1],0,3);
umod=tfd2mod(T,2,g11,g21,g12,g22);
%定义扰动模型
g13=poly2tfd(3.8,[14.9 1],0,8);
```

```
g23=poly2tfd(4.9,[13.2 1],0,3);
dmod=tfd2mod(T,2,g13,g23);
%建立叠加了扰动的混合系统模型
pmod=addumd(umod,dmod);
%考虑精确建模的情况
imod=pmod;
ywt=[];
uwt=[];
%预测时域和控制时域均为 5
P=5;
M=P;
Ks=smpccon(imod,ywt,uwt,M,P);
%仿真时间为 30s
tend=30;
r=[1 0];
[y,u]=smpcsim(pmod,imod,Ks,tend,r);
plotall(y,u,T)
```

在精确建模的情况下，闭环系统的输出曲线和控制量变化曲线如图 4.4.5 和图 4.4.6 所示。

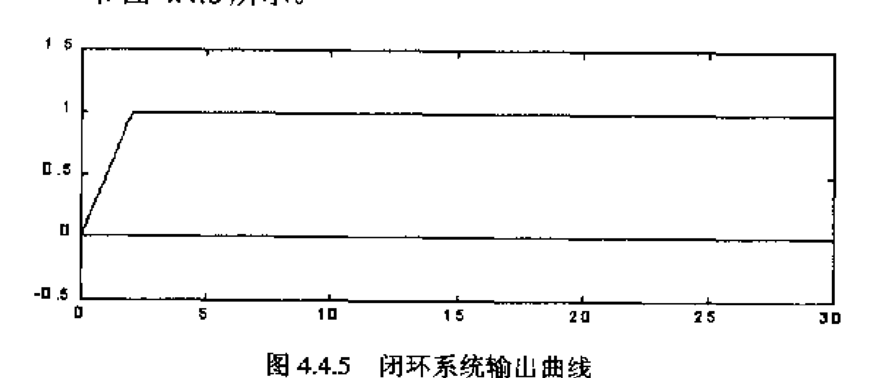


图 4.4.5　闭环系统输出曲线

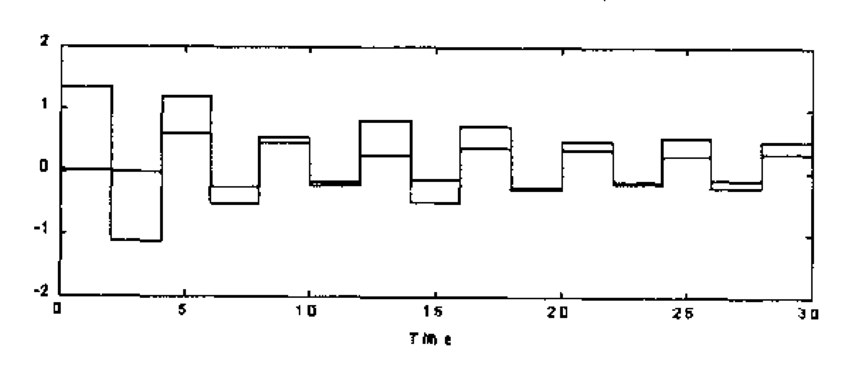


图 4.4.6　控制量变化曲线

%增加预测时域长度，同时减少控制时域长度，闭环系统的输出曲线和控制量变化曲线分别如图 4.4.7 和图 4.4.8 所示。

```
%预测时域长度为 10
  P=10;
%控制时域长度为 3
  M=3;
  Ks=smpccon(imod,ywt,uwt,M,P);
  [y,u]=smpcsim(pmod,imod,Ks,tend,r);
%绘制输入输出曲线
  plotall(y,u,T)
```

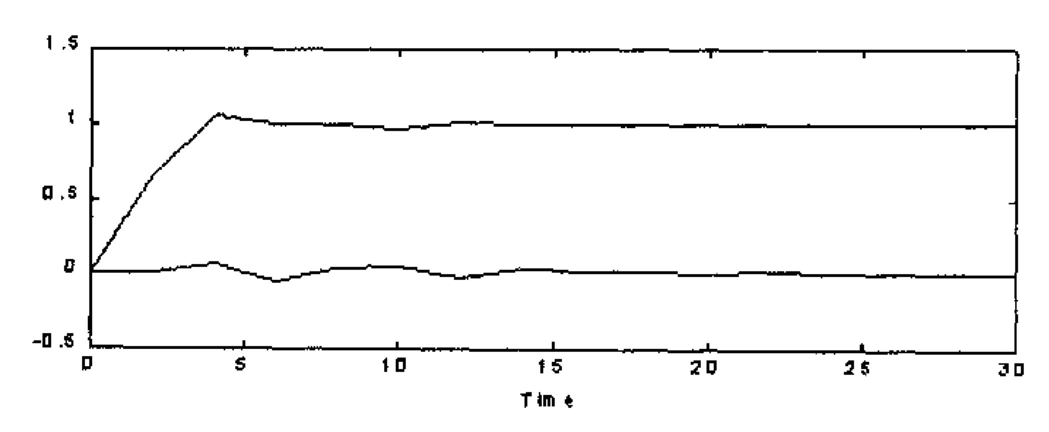


图 4.4.7　改变预测时域长度后闭环系统响应

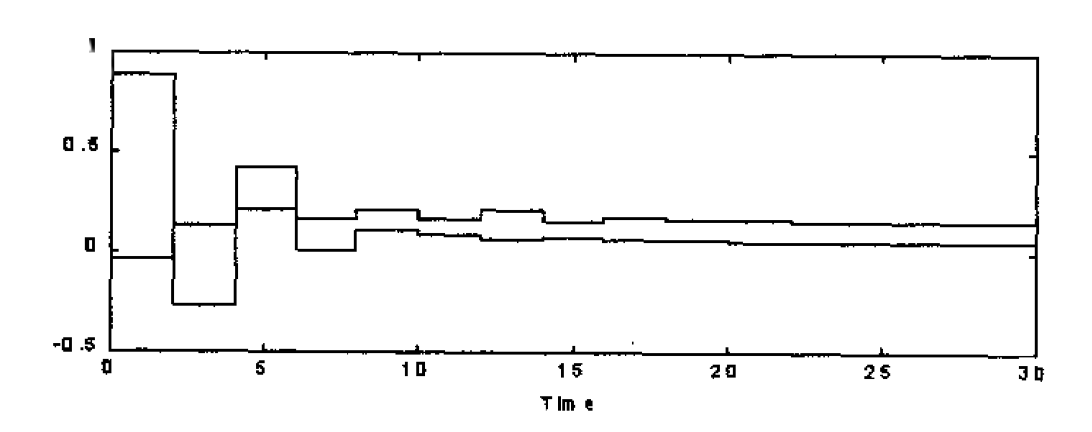


图 4.4.8　改变预测时域长度后的控制量变化曲线

%进一步改变控制时域长度，采用控制量分块的形式，闭环系统的输出曲线和控制量变化曲线分别如图 4.4.9 和图 4.4.10 所示。

```
M=[2 3 4];
Ks=smpccon(imod,ywt,uwt,M,P);
[y,u]=smpcsim(pmod,imod,Ks,tend,r);
plotall(y,u,T)
```

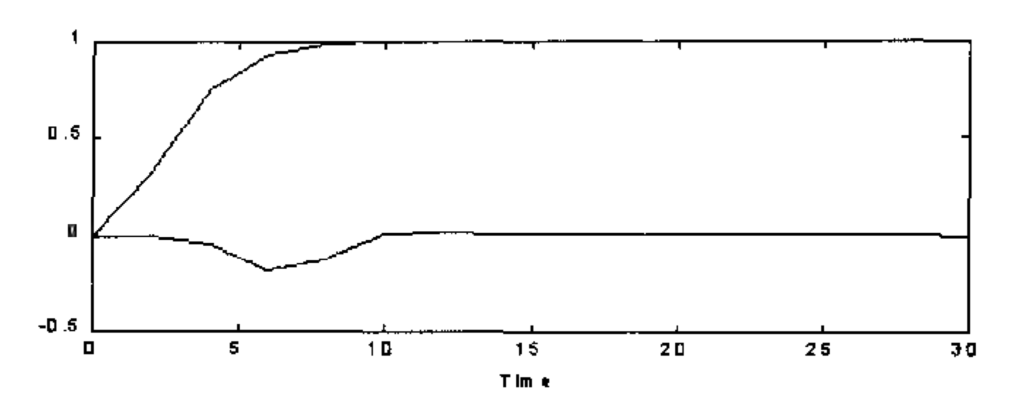


图 4.4.9　改变控制时域后的系统响应

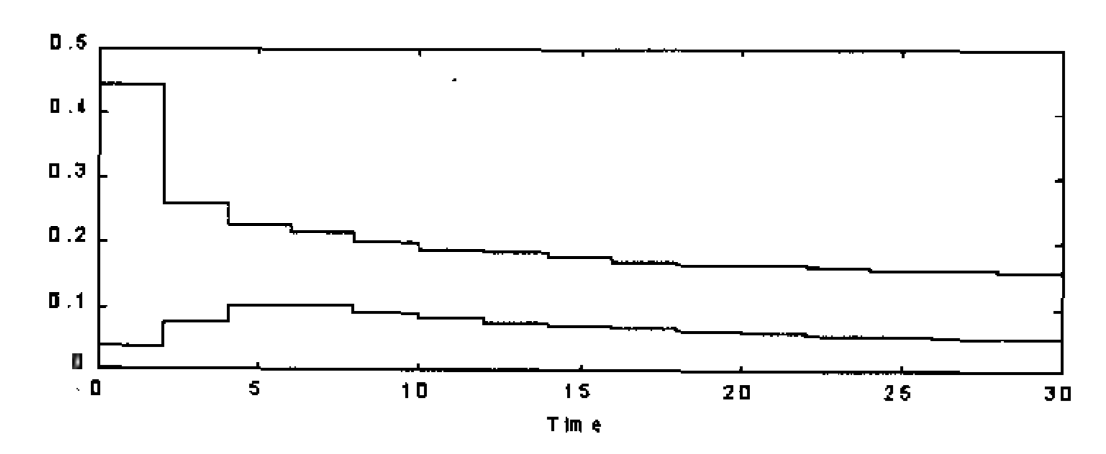


图 4.4.10　控制时域改变后的控制量变化曲线

```
%增加输入控制量的加权矩阵系数，模型预测闭环控制系统的响应曲线和控制量变化曲线分别如图 4.4.11 和 4.4.12 所示。
uwt=[1 1];
%预测时域长度为 5
P=5;
%控制时域长度等于预测时域长度
M=P;
Ks=smpccon(imod,ywt,uwt,M,P);
[y,u]=smpcsim(pmod,imod,Ks,tend,r);
plotall(y,u,T)
%将输出设定值均设为 0，绘制闭环系统的输出响应和控制量变化曲线，如图 4.4.13 和 4.4.14 所示。
ulim=[];
Kest=[];
r=[0 0];
%测量噪声和扰动为 0
z=[];
```

```
v=[];
w=[1];
[y,u]=smpcsim(pmod,imod,Ks,tend,r,ulim,Kest,z,v,w);
plotall(y,u,T)
```

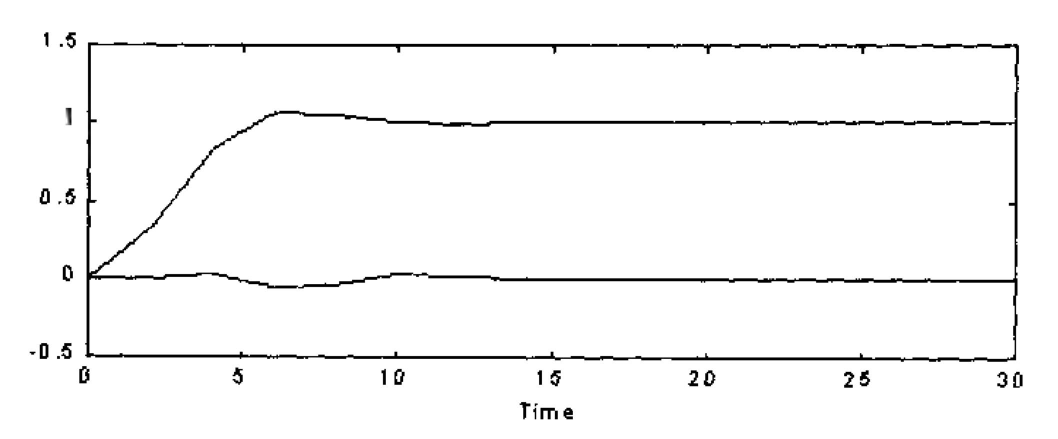


图 4.4.11　增加控制量加权矩阵后的输出响应

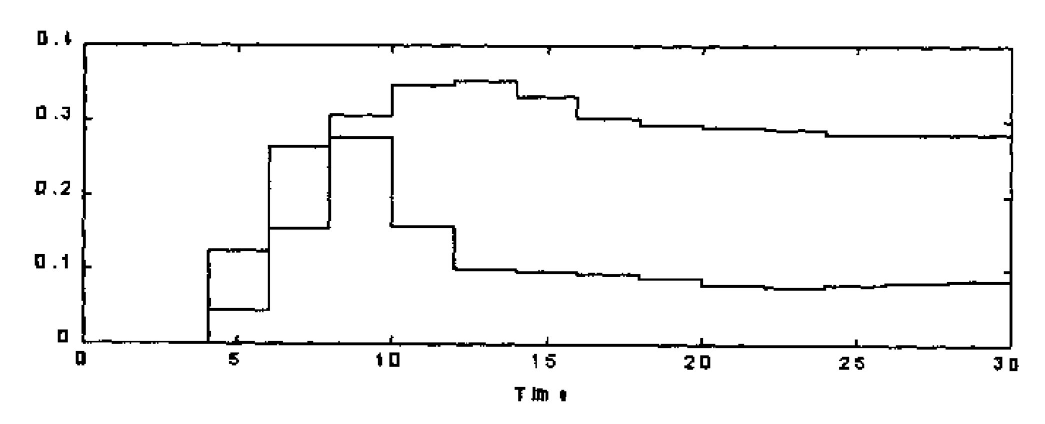


图 4.4.12　增加控制量加权矩阵后的控制量变化曲线

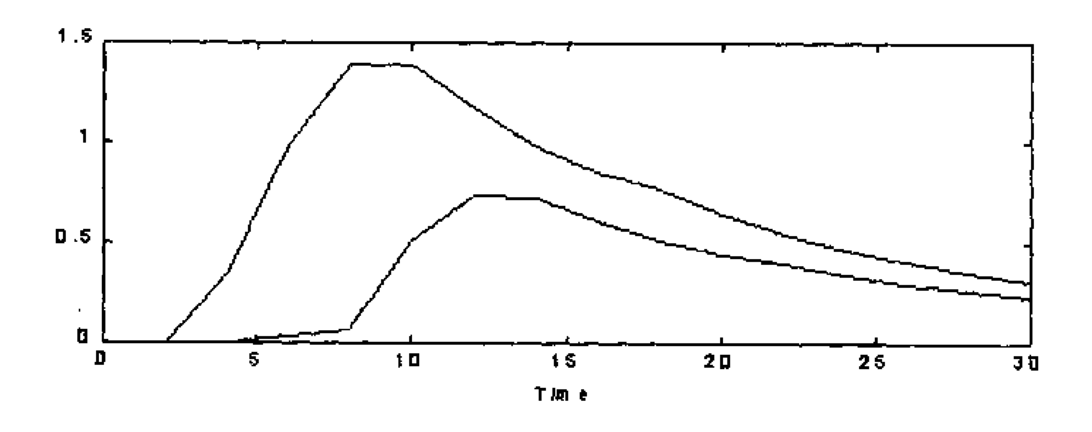


图 4.4.13　零设定值时闭环系统的输出响应

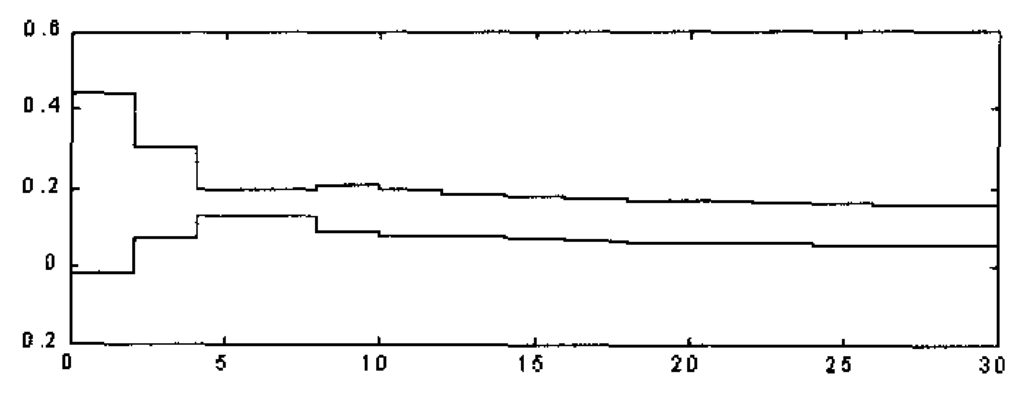


图 4.4.14　零设定值时的控制量变化曲线

%采用估计器进一步改善系统性能（参见函数 smpcest），对应的系统输出和控制量变化曲线如图 4.4.15 和 4.4.16 所示。

```
[Kest,newmod]=smpcest(imod,[15 15],[3 3]);
Ks=smpccon(newmod,ywt,uwt,M,P);
[y,u]=smpcsim(pmod,newmod,Ks,tend,r,ulim,Kest,z,v,w)
;
plotall(y,u,T)
```

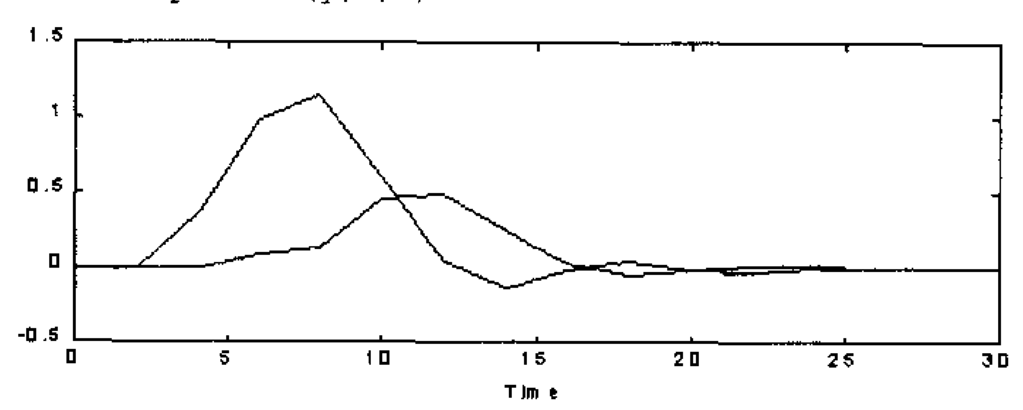


图 4.4.15　增加估计器后的系统响应曲线

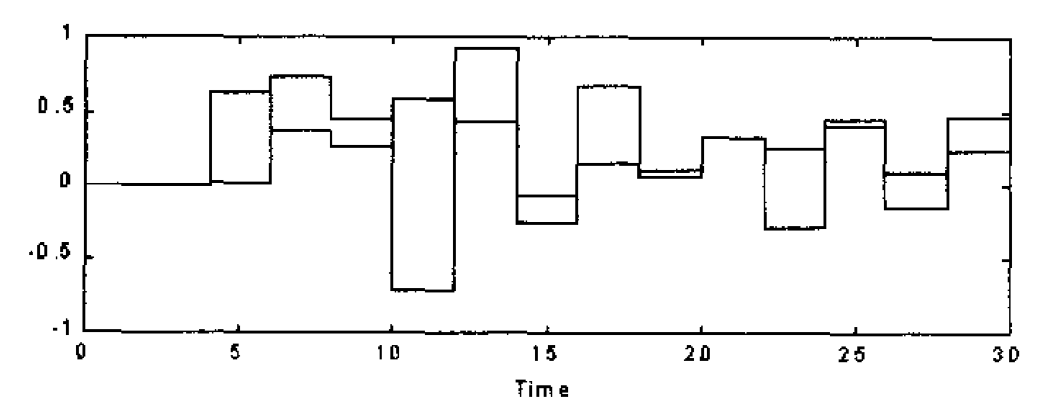


图 4.4.16　增加估计器后的控制量变化曲线

4.4.3 状态估计器设计

函数 smpcest 用于系统的状态估计器设计，该函数的使用方法说明如下。

- **smpcest**

功能：状态估计器设计。

格式：[Kest,newmod]=smpcest(imod,tau,signoise)

Kest=smpcest(imod,Q,R)

说明：输入参数定义为：

imod——系统的内部模型；

tau——时间常数向量，tau 的各个分量用于指定扰动对每个输出的影响特性；

signoise——每个输出的信噪比；

Q——未测量扰动的方差矩阵；

R——测量噪声的方差矩阵。

输出参数定义为：

Kest——状态估计器的增益矩阵；

newmodel——修改了的系统模型，在该模型中引入了新的状态来表示扰动的影响。

举例：仍然考虑如下的多变量系统：

$$\begin{bmatrix} \dfrac{12.8e^{-s}}{16.7s+1} & \dfrac{6.6e^{-7s}}{10.9s+1} \\ \dfrac{-18.9e^{-3s}}{21s+1} & \dfrac{-19.4e^{-3s}}{14.4s+1} \end{bmatrix}$$

```
g11=poly2tfd(12.8,[16.7 1],0,1);
g21=poly2tfd(6.6,[10.9 1],0,7);
g12=poly2tfd(-18.9,[21.0 1],0,3);
g22=poly2tfd(-19.4,[14.4 1],0,3);
delt=1; ny=2;
imod=tfd2mod(delt,ny,g11,g21,g12,g22);
gw1=poly2tfd(3.8,[14.9 1],0,8);
gw2=poly2tfd(4.9,[13.2 1],0,3);
pmod=addumd(imod,tfd2mod(delt,ny,gw1,gw2));
%设计模型预测控制器
P=6;    % Prediction horizon.
M=2;    % Number of moves (input horizon).
ywt=[]; uwt=[1 1];
Ks=smpccon(imod,ywt,uwt,M,P);
```

```
[y3,u3]=smpcsim(pmod,imod,Ks,tend,r,ulim,[],z,v,w,wu);
%设计状态估计器
Kest1=smpcest(pmod,1,0.001*eye(ny));
Ks1=smpccon(pmod,ywt,uwt,M,P);
r=[];
ulim=[]; z=[]; v=[]; w=[1]; wu=[]; tend=30;
[y1,u1]=smpcsim(pmod,pmod,Ks1,tend,r,ulim,Kest1,z,v,w,w
u)
plotall(y1,u1,delt)
```

采用了状态估计器的闭环系统输出响应和控制量曲线如图 4.4.17 和 4.4.18 所示。

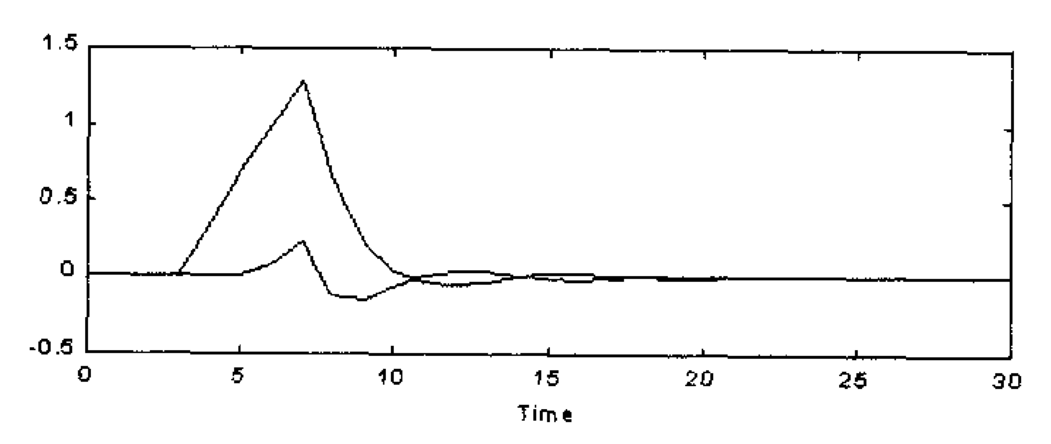


图 4.4.17　系统输出响应曲线

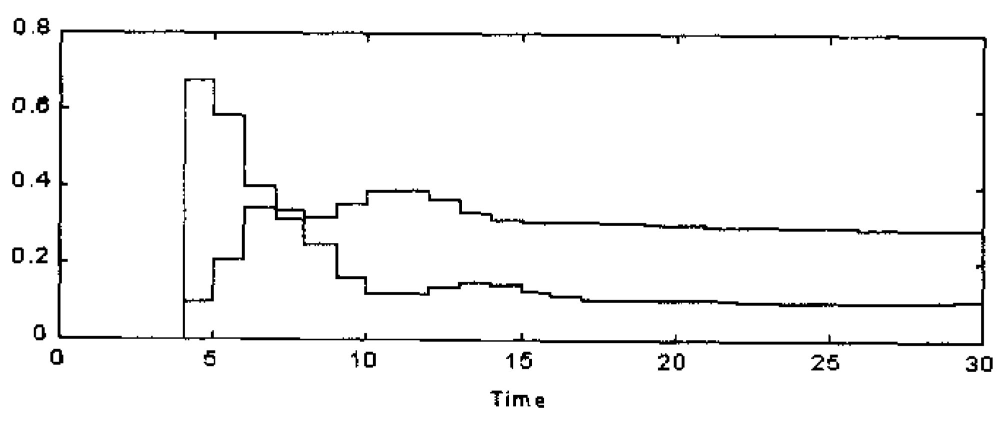


图 4.4.18　控制量曲线

```
% 下面进行简化的状态估计器设计
tau=[10 10];
signoise=[3 3];
[Kest2,newmod]=smpcest(imod,tau,signoise);
Ks2=smpccon(newmod,ywt,uwt,M,P);
```

```
[y2,u2]=smpcsim(pmod,newmod,Ks2,tend,r,ulim,Kest2,z,v,w,wu);
```

采用简化的状态估计器的闭环系统输出响应曲线和控制量曲线如图 4.4.19 和 4.4.20 所示。

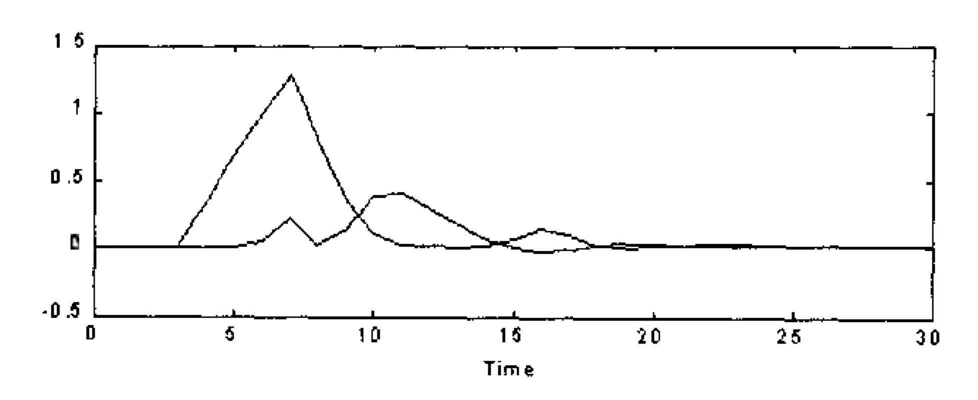


图 4.4.19 简化状态估计器设计后的系统输出响应曲线

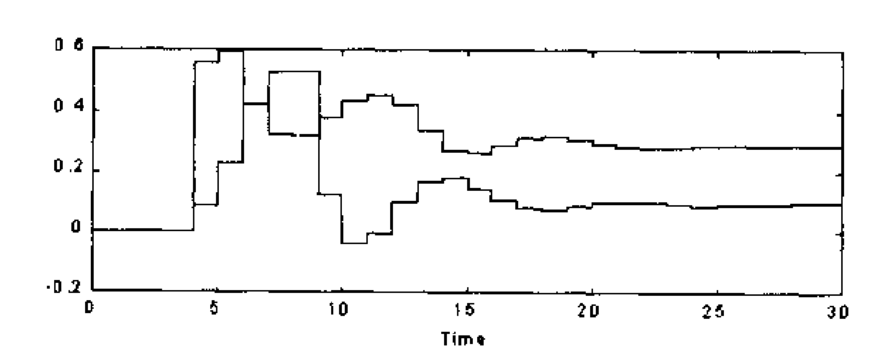


图 4.4.20 简化状态估计器设计后的的控制量曲线

4.5 系统分析与绘图函数

前面介绍了模型预测控制工具箱的系统设计与仿真功能函数，这些函数提供了进行模型预测控制器设计的有力工具。为进一步完善其功能，模型预测工具箱还包括若干系统分析和绘图的功能函数，如表 4.5.1 所示。

表 4.5.1 系统分析与绘图函数

函数名称	功能
mod2frsp	计算系统（MPC 状态空间模型）的频率响应
smpcgain	计算系统（MPC 状态空间模型）的稳态增益矩阵
smpcpole	计算系统（MPC 状态空间模型）的极点
svdfrsp	计算频率响应的奇异值
mpcinfo	输出系统表示的矩阵类型和属性
plotall	绘制系统仿真的输入输出曲线（一个图形窗口）
plotfrsp	绘制系统的频率响应波特图
ploteach	在多个图形窗口分别绘制系统的输入输出仿真曲线
plotstep	绘制系统阶跃响应模型的曲线

4.5.1　计算和绘制系统的频率响应曲线

函数 mod2frsp 和 plotfrsp 分别用于计算和绘制系统的频率响应特性，使用说明如下。

1　mod2frsp

功能：计算 MPC 状态空间模型系统的频率响应。

格式：[frsp,eyefrsp] = mod2frsp(mod,freq,out,in,balflg)

说明：输入参数定义为：

mod——系统的 MPC mod 模型；

freq——频率向量，指定对数频率的区间范围和频率点个数；

out——指定输出变量，如果 out=[ ]，则计算所有输出；

in——指定输入变量，如果 in=[ ]，则计算所有输入；

balflag——如果 balflag 为非零值，则在计算之前进行 PHI 矩阵的均衡处理。

输出参数定义为：

frsp——系统的输出频率响应矩阵；

eyefrsp——当频率响应矩阵为方阵时，eyefrsp=I－frsp，其中 I 为单位对角矩阵。

2　plotfrsp

功能：绘制系统的频率响应波特图。

格式：plotfrsp(vmat)

plotfrsp(vmat,out,in)

说明：输入参数定义为：

vmat——系统的频率响应矩阵；

out——指定输出变量；

in——指定输入变量。

举例：计算并绘制系统的频率响应曲线，如图 4.5.1 所示。设系统的传递函数为

$$G(s) = \frac{3e^{-4s}}{5s+1}$$

```
g=poly2tfd(3,[5 1],0,4)
mod=tfd2mod(0.1, 1,g)
vmat=mod2frsp(mod,[-2,3,50],1,1,0)
plotfrsp(vmat)
```

4.5.2　计算频率响应的奇异值

在获得了系统的频率响应矩阵后，函数 svdfrsp 用于计算系统频率响应矩阵的奇异值，

其使用方法如下。

- **svdfrsp**

功能：计算系统频率响应矩阵的奇异值。

格式：[sigma,omega]=svdfrsp(vmat)

说明：输入参数定义为：

vmat——系统的频率响应矩阵。

输出参数定义为：

sigma——频率响应的奇异值矩阵。sigma 的第 I 行为第 I 个频率响应子矩阵的按降序排列的奇异值；

omega——包含频率响应矩阵的独立变量值的列向量。

举例：

```
g=poly2tfd(3,[5 1],0,4)
mod=tfd2mod(0.1, 1,g)
vmat=mod2frsp(mod,[-2,3,50],1,1,0)
plotfrsp(vmat)
```

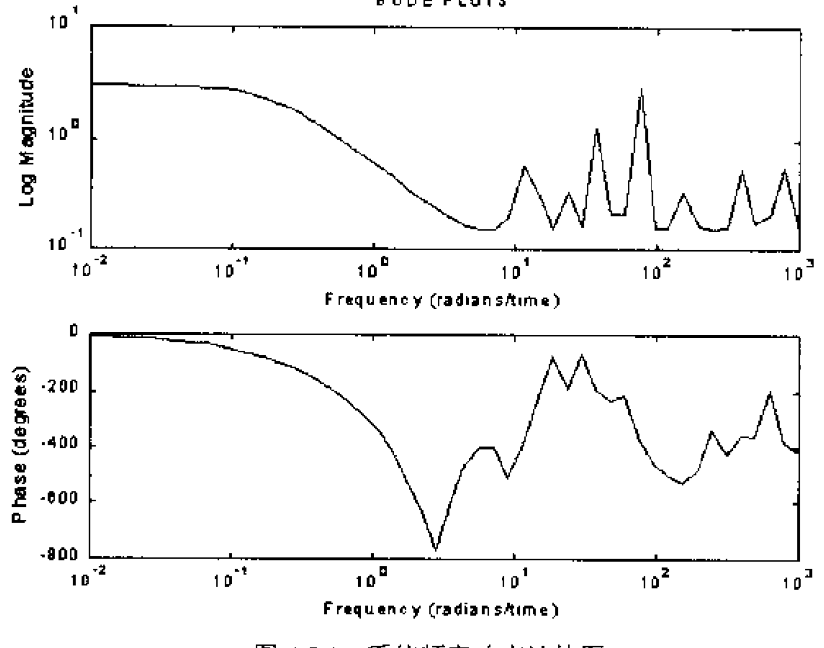


图 4.5.1　系统频率响应波特图

4.5.3　计算系统的极点和稳态增益矩阵

1　smpcpole

功能：计算系统的极点。

格式：[poles]=smpcpole(mod)

说明：输入参数定义为：

mod——系统的 MPC mod 模型。

输出参数定义为：

poles —以复数向量给出的系统的极点。

2 smpcgain

功能：计算系统的稳态增益矩阵。

格式：g=smpcgain(mod)

说明：输入参数定义为：

mod——开环稳定的对象模型（MPC mod 格式）。

输出参数定义为：

g——系统的稳态增益矩阵，该矩阵的行对应输出变量，列对应输入变量。

举例：

```
g=poly2tfd(2,[4 1],0,3)
mod=tfd2mod(0.1, 1,g)
[poles]=smpcpole(mod)
g=smpcgain(mod)
```

4.5.4 其他系统分析和绘图函数

函数 mpcinfo 用于获得有关系统模型矩阵的信息，其用法说明如下。

1 mpcinfo

功能：返回系统模型矩阵的信息。

格式：flag=mpcinfo(mat)

说明：输入参数定义为：

mat——系统模型矩阵。

输出参数定义为：

flag——矩阵的类型，包括

- flag<0——常数矩阵；
- flag=1——系统矩阵；
- flag=2——MPC 状态空间模型；
- flag=3——MPC 阶跃响应模型。

举例：

```
g=poly2tfd(2,[4 1],0,3)
mod=tfd2mod(0.1, 1,g)
flag1=mpcinfo(g)
flag2=mpcinfo(mod)
```

输出结果为：

```
    flag1=-3
```

```
flag2=4  %MPC 状态空间模型。
```

2 plotall

功能：绘制系统仿真的输入输出曲线。

格式：plotall(y,u)

plotall(y,u,t)

说明：输入参数定义为：

y,u——系统的输入输出变量，其中控制量 u 在绘图之前转换为阶梯型连续数据变量；

t——采用周期。

3 ploteach

功能：在多个窗口绘制系统仿真的输入输出曲线。

格式：ploteach(y)

ploteach(y,u,t)

ploteach(y,u)

ploteach([],u)

ploteach(y,[],t)

说明：输入参数定义与函数 plotall 相同。

举例：

```
g=poly2tfd(2,[4 1],0,3)
mod=tfd2mod(2, 1,g)
P=7;    % Prediction horizon.
M=4;    % Number of moves (input horizon).
ywt=[ ];
uwt=1 ;
tend=30;
r=2;
ulim=[-inf  inf 100];
ylim=[];
delt=2
[y,u]=scmpc(mod,mod,ywt,uwt,M,P,tend,r,ulim,ylim);
ploteach(y,u,delt)
```

4 plotstep

功能：绘制系统阶跃响应模型的曲线。

格式：plotstep(plant)

plotstep(plant,opt)

说明：输入参数定义为：

plant——对象的阶跃响应模型；

opt——指定输出变量。

举例：绘制系统的阶跃响应曲线，如图 4.5.2 所示。

```
g=poly2tfd(2,[4 1],0,3)
plant=tfd2step(30,2,1,g)
plotstep(plant)
```

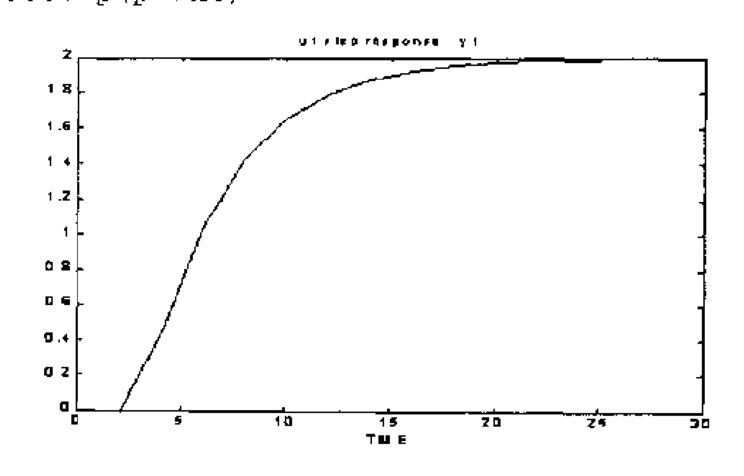


图 4.5.2　系统阶跃响应曲线

4.6　模型预测工具箱的通用功能函数

模型预测工具箱的通用功能函数包括模型转换函数、离散系统分析于仿真函数和方程求解函数等，如表 4.6.1 所示。

表 4.6.1　模型预测工具箱的通用功能函数

函数名称	功　能
abcdchkm	检查状态空间矩阵 A，B，C，D 的维数一致性
cp2dp	将连续型多项式传递函数转换为离散型多项式传递函数
c2dmp	连续系统离散化
dantzgmp	求解二次规划问题
dareiter	求解离散 Riccati 方程
dimpulsm	生成离散系统的脉冲响应
dlqe2	计算离散系统的状态估计器增益矩阵
dlsimm	离散系统仿真
d2cmp	将离散系统转换为连续函数
mpcaugss	增广状态空间模型
mpcparal	将两个状态空间模型并联
mpcstair	生成阶梯型控制变量
parpart	分割参数用于 Simulink 仿真
ss2tf2	将状态空间模型转换为传递函数
tf2ssm	就传递函数转换为状态空间模型

4.6.1　通用模型转换函数

1　cp2dp

功能：将连续型多项式传递函数转换为离散型多项式传递函数。

格式：[numd,dend]=cp2dp(num,den,delt,delay)

说明：输入参数定义为：

num——连续传递函数的分子多项式；

den——连续传递函数的分母多项式；

delt——采样周期；

delay——系统时延。

输出参数定义为：

numd——离散传递函数的分子多项式；

dend——离散传递函数的分母多项式。

2 c2dmp

功能：连续系统离散化。

格式：[Phi, Gam, f0] = c2dmp(A,B,T,dx0)

说明：输入参数定义为：

B——连续系统状态空间矩阵；

T——采样周期；

dx0——可选参数，缺省值为 0，用于不稳定条件下的模型线性化。

输出参数定义为：

Phi，Gam——离散系统的状态空间矩阵；

f0——对应 dx0 的离散化参数。

3 d2cmp

功能：离散系统转换为连续系统。

格式：[A,B]=d2cmp(Phi,Gam,T)

说明：输入参数定义为：

Phi，Gam——离散系统的状态空间矩阵；

T——采样周期。

输出参数定义为：

A，B——连续系统的状态空间矩阵。

4 mpcaugss

功能：增广状态空间模型。

格式：[phia,gama,ca]=mpcaugss(phi,gam,c)

[phia,gama,ca,da]=mpcaugss(phi,gam,c,d)

说明：输入参数定义为：

phi，gam，c，d——系统 MPC 状态空间模型矩阵。

输出参数定义为：

phia,gama,ca,da——增广系统的 MPC 状态空间矩阵。

5 mpcpara

功能：并联两个状态空间模型。

格式：[A,B,C,D] = MPCPARAL(A1,B1,C1,D1,A2,B2,C2,D2)

说明：输入参数定义为：

A1,B1,C1,D1——子系统 1 的状态空间矩阵；

A2,B2,C2,D2——子系统 2 的状态空间矩阵。

输出参数定义为：

A,B,C,D——并联系统的状态空间矩阵。

6 ss2tf2

功能：状态空间模型转换为传递函数模型。

格式：[Num,Den] = ss2tfs(A,B,C,D,iu)

说明：输入参数定义为：

A,B,C,D——系统状态空间矩阵；

Iu——指定输入变量。

输出参数定义为：

Num,Den——传递函数模型。

7 tf2ssm

功能：传递函数模型转换为状态空间模型。

格式：[A,B,C,D] = tf2ssm(Num, Den)

说明：输入参数定义为：

Num——传递函数的分子多项式系数向量；

Den——传递函数分母多项式系数向量。

输出参数定义为：

A,B,C,D——系统状态空间模型矩阵。

4.6.2 方程求解函数

1 dantzgmp

功能：求解二次规划问题。

格式：[bas,ib,il,iter,tab]=dantzgmp(tabi,basi,ibi,ili)

说明：输入参数中，basi 初始基向量；输出参数中，bas 为最终的基向量，iter 为迭代次数，il 为 Lagrange 乘子的索引向量。

2 dareiter

功能：求解离散 Riccati 方程。

格式：X = dareiter(F, G1, G2, H)

说明：输入参数定义为：

F, G1, G2, H——离散 Riccati 方程的系数矩阵，离散 Riccati 方程具有如下形式：

$$F^T XF - X - F^T XG_1(G_2 + G_1^T XG_1)^{-1} G_1^T XF + H = 0$$

输出参数定义为：

X——离散 Riccati 方程的解。

4.6.3 离散系统的分析函数

1 dlqe2

功能：计算离散系统的状态估计器增益矩阵。

格式：k=dlqe2(phi,gamw,c,q,r)

[k,m,p]=dlqe2(phi,gamw,c,q,r)

说明：函数 dlqe2 利用迭代方法求解 Kalman 滤波器方程。

输入参数定义如下：

phi，c——对象的 MPC 状态空间矩阵；

gamw——不可测扰动的特性矩阵；

q——不可测扰动的方差矩阵；

r——测量噪声方差矩阵。

输出参数定义为：

k——离散 Kalman 滤波器的稳态增益矩阵；

m——在测量值更新之前的状态估计期望方差；

p——在测量值更新之后的状态估计期望方差。

2 dlsimm

功能：离散系统仿真。

格式：[y,x] = dlsimm(A,B,C,D,u,x0);

[y,x] = dlsimm(num,den,u);

说明：输入参数定义为：

A，B，C，D——离散系统的状态空间模型；

num,den——离散系统的传递函数模型；

u——输入变量；

x0——初始条件。

输出参数定义为：

y——离散系统的输出响应；

x——离散系统的状态响应。

参考文献

[1] 舒迪前，预测控制系统及其应用，机械工业出版社，1996

[2] Matlab MPC Toolbox Demo,Mathworks Inc,1998

第5章 模糊逻辑工具箱

(Fuzzy Logic Toolbox,Ver 2)

"模糊"是与"精确"相对的概念。模糊性普遍存在于人类思维和语言交流中，是一种不确定性的表现。随机性则是客观存在的另一类不确定性，两者虽然都是不确定性，但存在本质的区别。模糊性主要是人对概念外延的的主观理解上的不确定性。随机性则主要反映客观上的自然的不确定性，即对事件或行为的发生与否的不确定性。

在人类科学的发展史上，长期以来，以追求事物的精确性描述为目的的研究产生了大量的成果，并建立了庞大的数学和逻辑学体系。随着人类科技的进步，在社会生产和生活中存在的大量的不确定性开始引起了人们的注意。以概率论的创立为标志，针对事物随机性的研究开始得到较快的发展。有关模糊不确定性的研究直到 1965 年美国教授 L.A.Zadeh 首次提出模糊集合的概念之后才得到广泛开展。目前已形成模糊逻辑和模糊数学这两门新兴的学科。

模糊逻辑和模糊数学虽然只有短短的三十余年历史，但其理论和应用的研究已取得了丰富的成果。尤其是随着模糊逻辑在自动控制领域的成功应用，模糊控制理论和方法的研究引起了学术界和工业界的广泛关注。在模糊理论研究方面，以 Zadeh 提出的分解定理和表现定理为基础的模糊数学理论已有大量的成果问世。1984 年成立了国际模糊系统协会（IFSA），FUZZY SETS AND SYSTEMS （模糊集与系统）杂志与 IEEE（美国电气与电子工程师协会）"模糊系统"杂志也先后创刊。在模糊逻辑的应用方面，自从 1974 年英国的 Mamdani 首次将模糊逻辑用于蒸汽机的控制后，模糊控制在工业过程控制、机器人、交通运输等方面得到了广泛而卓有成效的应用。与传统控制方法如 PID 控制相比，模糊控制利用人类专家控制经验，对于非线性、复杂对象的控制显示了鲁棒性好、控制性能高的优点。模糊逻辑的其他应用领域包括：聚类分析、故障诊断、专家系统和图像识别等。

目前，模糊逻辑与神经网络、进化计算一起成为计算智能这一新兴学科的三大组成部分。IEEE 定期举行国际计算智能大会，国际上正逐渐形成以计算智能为主要研究内容的软计算领域。模糊逻辑理论与应用的发展将会极大地推动人类社会向智能信息处理的新阶段迈进。

针对模糊逻辑尤其是模糊控制的迅速推广应用，MathWorks 公司在其 Matlab 版中添加了 Fuzzy Logic 工具箱。该工具箱由长期从事模糊逻辑和模糊控制研究与开发工作的有关专家和技术人员编制，并在 Matlab 5.1 版中推出了 2.0 版本。Matlab Fuzzy Logic 工具箱以其功能强大和方便易用的特点得到了用户的广泛欢迎。模糊逻辑的创始人 Zadeh 教授称赞该工具箱"在各方面都给人以深刻的印象，使模糊逻辑成为智能系统的概念与设计的有效工具。"

概括来说，Matlab Fuzzy Logic 工具箱具有如下 5 个方面的功能特点。

1　易于使用

模糊逻辑工具箱提供了建立和测试模糊逻辑系统的一整套功能函数，包括定义语言变量及其隶属度函数、输入模糊推理规则、对整个模糊推理系统的管理以及交互式地观察模糊推理的过程和输出结果。

2　提供图形化的系统设计界面

在模糊逻辑工具箱中包含五个图形化的系统设计工具，这五个设计工具是：

- 模糊推理系统编辑器，该编辑器用于建立模糊逻辑系统的整体框架，包括输入与输出数目、去模糊化方法等；
- 隶属度函数编辑器，用于通过可视化手段建立语言变量的隶属度函数；
- 模糊推理规则编辑器；
- 系统输入输出特性曲面浏览器；
- 模糊推理过程浏览器。

3　支持模糊逻辑中的高级技术

- 自适应神经-模糊推理系统（ANFIS，Adaptive Neural - Fuzzy Inference System）；
- 用于模式识别的模糊聚类技术；
- 模糊推理方法的选择，用户可在广泛采用的 Mamdani 型推理方法和 Sugeno 型推理方法两者之间选择。

4　集成的仿真和代码生成功能

模糊逻辑工具箱不但能够实现与 Simulink 的无缝连接，而且通过 Real-Time Workshop2.1 能够生成 ANSI C 源代码，从而易于实现模糊系统的实时应用。

5　独立运行的模糊推理机。

在用户完成模糊逻辑系统的设计后，可以将设计结果以 ASCII 码文件保存；利用模糊逻辑工具箱提供的模糊推理机，可以实现模糊逻辑系统的独立运行或者作为其他应用的一部分运行。

5.1　模糊集合与模糊关系

模糊数学的研究为模糊推理系统的应用奠定了理论基础，并为其分析和设计提供理论上的指导。开展模糊数学的研究对推广模糊推理系统尤其是模糊控制的应用具有重要的意义。自从 1965 年 Zadeh 首次提出模糊集合的概念以来，模糊数学理论取得了一系列的研究成果。在讨论模糊推理系统的分析与设计之前，本节将简要介绍模糊数学中与模糊推理系统应用有密切关系的概念和结论。

5.1.1 模糊集合

1 经典集合及其特征函数

经典集合的概念是由 19 世纪末德国数学家 G.Contor 建立的，Contor 创立的集合论已成为现代数学的基础。在经典集合论中，集合被定义为具有某种属性的、确定的、彼此间可以区别的事物的全体。组成集合的事物称为集合的元素或元。经典集合的元素与集合的关系可用特征函数来表达。某一集合 A 的特征函数定义如下：

$$f_A(x)=\begin{cases}1, & x\in A\\0, & x\notin A\end{cases}$$

从中可以看出，一个元素与集合的关系只有两种，即或者属于集合 A，或者不属于集合 A。

2 模糊集合及其隶属度函数

用上述经典集合的定义所表达的概念内涵和外延都是明确的，但在人们的事物和语言表达中，有许多没有明确外延的概念，即模糊概念，如“高个子”、“青年”等。模糊概念无法用经典集合加以描述。为解决这一问题，Zadeh 于 1965 年在其著名的论文“FUZZY SETS ”中给出了模糊集合的定义。

定义 5.1 设给定论域 U，U 到[0，1]闭区间的任一映射 μ_A ：

$$\mu_A:\quad U\to[0,1]$$

$$u\to\mu_{\underset{\sim}{A}}(u)$$

都确定 U 的一个模糊子集，映射 $\mu_{\underset{\sim}{A}}(u)$ 称为模糊子集 $\underset{\sim}{A}$ 的隶属度函数，$\mu_{\underset{\sim}{A}}(u)$ 称为 u 对于模糊集合 $\underset{\sim}{A}$ 的隶属度，隶属度也可记为 $\underset{\sim}{A}(u)$ 。在不引起混淆的情况下，模糊子集也称为模糊集合。

上述定义表明，论域 U 上的模糊子集 $\underset{\sim}{A}$ 由隶属度函数 $\mu_{\underset{\sim}{A}}(u)$ 来表征，$\mu_{\underset{\sim}{A}}(u)$ 的取值范围为闭区间[0，1]，$\mu_{\underset{\sim}{A}}(u)$ 的大小反映了 u 对于模糊子集 $\underset{\sim}{A}$ 的隶属程度。

3 模糊集合与经典集合的关系

为说明模糊集与经典集合的联系，先引入截集的概念。

定义 5.2：λ 截集 设 $\underset{\sim}{A}$ 是论域 U 上的模糊子集，称

$$\underset{\sim}{A}_\lambda=\{u,\mid\mu_{\underset{\sim}{A}}(u)\geq\lambda,0\leq\lambda\leq1\}$$

为 $\underset{\sim}{A}$ 的 λ 截集，它是一个经典集合，λ 称为水平。

定义 5.3：λ 强截集　设 $\underset{\sim}{A}$ 是论域 U 上的模糊子集，称

$$\underset{\sim}{A}_{\lambda}=\{u,|\mu_{\underset{\sim}{A}}(u)>\lambda,0\le\lambda\le 1\}$$

为 $\underset{\sim}{A}$ 的 λ 强截集，它也是一个经典集合。

在定义了截集的概念之后，下面的分解定理和扩张原理建立了模糊集合与经典集合的关系，并构成了模糊数学的基础。

定理 5.1:分解定理　设 $\underset{\sim}{A}$ 为论域 U 上的模糊集合，A_λ 是 $\underset{\sim}{A}$ 的 λ 截集，$\lambda\in[0,1]$,则有如下的分解式成立：

$$\underset{\sim}{A}=\bigcup_{\lambda\in[0,1]}\lambda A_\lambda$$

其中，λA_λ 表示语言变量 X 的一个模糊子集，称为 λ 与 A_λ 的“乘积”，其隶属度函数定义为

$$\mu_{\lambda A_\lambda}(x)=\begin{cases}\lambda\ , & x\in A_\lambda\\ 0, & x\notin A_\lambda\end{cases}$$

或

$$\lambda A_\lambda=\lambda\wedge A_\lambda,\quad \text{“}\wedge\text{” 为取极小运算。}$$

扩张原理　设 U 和 V 是两个论域，f 是 U 到 V 的一个映射，对 U 上的模糊集合 $\underset{\sim}{A}$，由下式在 V 上定义一个模糊集合 $\underset{\sim}{B}$：

$$\mu_{\underset{\sim}{B}}(\mathrm{v})=\sup_{u\in f^{-1}(v)}[\mu_{\underset{\sim}{A}}(u)]$$

当 $f^{-1}(v)$ 对某些 $v\in V$ 为空集时，$\mu_{\underset{\sim}{B}}(v)=0$。

4　模糊集合的运算

与经典集合运算类似，模糊集合之间也存在交、并、补等运算关系。设 $\underset{\sim}{A}$，$\underset{\sim}{B}$ 是论域 U 上的模糊集合，$\underset{\sim}{A}$ 与 $\underset{\sim}{B}$ 的交集 $\underset{\sim}{A}\cap\underset{\sim}{B}$、并集 $\underset{\sim}{A}\cup\underset{\sim}{B}$ 和 $\underset{\sim}{A}$ 的补集 $\bar{\underset{\sim}{A}}$ 也是论域 U 上的模糊集合。设任意元素 $u\in U$，则 u 对 $\underset{\sim}{A}$ 与 $\underset{\sim}{B}$ 的交集、并集和 $\underset{\sim}{A}$ 的补集的隶属度函数分别定义如下。

定义 5.3：交运算　$\mu_{\underset{\sim}{A}\cap\underset{\sim}{B}}(u)=\min\{\mu_{\underset{\sim}{A}}(u),\mu_{\underset{\sim}{B}}(u)\}$

定义 5.4：并运算　$\mu_{\underset{\sim}{A}\cup\underset{\sim}{B}}(u)=\max\{\mu_{\underset{\sim}{A}}(u),\mu_{\underset{\sim}{B}}(u)\}$

定义 5.5：补运算　$\mu_{\bar{\underset{\sim}{A}}}=1-\mu_{\underset{\sim}{A}}(u)$

模糊集合的运算满足一系列性质，其中大部分与经典集合运算有类似的形式。下面列出了模糊集合运算的一些基本性质：

- 幂等律

$$\underset{\sim}{A}\cup\underset{\sim}{A}=\underset{\sim}{A}，\underset{\sim}{A}\cap\underset{\sim}{A}=\underset{\sim}{A}$$

- 交换律

$$\underset{\sim}{A}\cap\underset{\sim}{B}=\underset{\sim}{B}\cap\underset{\sim}{A}，\underset{\sim}{A}\cup\underset{\sim}{B}=\underset{\sim}{B}\cup\underset{\sim}{A}$$

- 结合律

$$(\underset{\sim}{A}\cup\underset{\sim}{B})\cup\underset{\sim}{C}=\underset{\sim}{A}\cup(\underset{\sim}{B}\cup\underset{\sim}{C})$$

$$(\underset{\sim}{A}\cap\underset{\sim}{B})\cap\underset{\sim}{C}=\underset{\sim}{A}\cap(\underset{\sim}{B}\cap\underset{\sim}{C})$$

- 分配律

$$\underset{\sim}{A}\cap(\underset{\sim}{B}\cup\underset{\sim}{C})=(\underset{\sim}{A}\cap\underset{\sim}{B})\cup(\underset{\sim}{A}\cap\underset{\sim}{C})$$

$$\underset{\sim}{A}\cup(\underset{\sim}{B}\cap\underset{\sim}{C})=(\underset{\sim}{A}\cup\underset{\sim}{B})\cap(\underset{\sim}{A}\cup\underset{\sim}{C})$$

- 吸收律

$$\underset{\sim}{A}\cap(\underset{\sim}{A}\cup\underset{\sim}{B})=\underset{\sim}{A}，\underset{\sim}{A}\cup(\underset{\sim}{A}\cap\underset{\sim}{B})=\underset{\sim}{A}$$

- 两极律

$$\underset{\sim}{A}\cap X=\underset{\sim}{A},\underset{\sim}{A}\cup X=\underset{\sim}{A}$$

$$\underset{\sim}{A}\cap\phi=\phi，\underset{\sim}{A}\cup\phi=\underset{\sim}{A}$$

- 复原律

$$\overline{\overline{\underset{\sim}{A}}}=\underset{\sim}{A}$$

- 德・摩根律

$$\overline{\underset{\sim}{A}\cup\underset{\sim}{B}}=\overline{\underset{\sim}{A}}\cap\overline{\underset{\sim}{B}},\overline{\underset{\sim}{A}\cap\underset{\sim}{B}}=\overline{\underset{\sim}{A}}\cup\overline{\underset{\sim}{B}}$$

与经典集合的运算性质比较，模糊集合的运算不满足排中律。经典集合的排中律运算性质如下：

$$A\cup\overline{A}=U,A\cap\overline{A}=\phi$$

这是由于模糊集合概念本身就是对经典集合非此即彼的排中律的一种突破。

5.1.2 模糊关系

经典关系是明确的关系，而模糊关系可看作经典关系的推广。下面在给出经典关系定义的基础上，引入模糊关系的定义。

定义 1：直积　设有两个经典集合 X 和 Y，定义 X 和 Y 的直积为：

$$X \times Y = \{(x, y) \mid x \in X, y \in Y\}$$

对 n 个集合 X_1，X_2，…X_n，直积 $X_1 \times X_2 \times \ldots \times X_n$ 定义为

$$X_1 \times X_2 \times \cdots \times X_n = \{(x_1, x_2, \cdots, x_n) \mid x_i \in X_i, i = 1,2,\cdots,n\}$$

定义 2：经典关系　对两个经典集合 X 和 Y，直集 $X \times Y$ 的子集 R 称为 X 到 Y 的一个二元关系，简称为关系。

n 个集合的直积 $X_1 \times X_2 \times \ldots \times X_n$ 上的关系则定义为 $X_1 \times X_2 \times \ldots \times X_n$ 的一个子集 R，称为 n 元关系。

在有限集的情况下，经典关系 R 可以直观地用布尔矩阵表示，布尔矩阵即元素均为 1 或 0 的矩阵。

设经典集合 $A = \{1,3,5\}$，$B=\{2,4,6\}$，则 A 到 B 的“>”关系 R 对应的关系矩阵为：

$$R = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

在给出经典关系的定义后，下面以类似的形式给出模糊关系的定义。

定义 3：模糊关系　直积空间 $X \times Y = \{(x, y) \mid x \in X, y \in Y\}$ 上的模糊关系是 $X \times Y$ 的一个模糊子集 $\underset{\sim}{R}$，$\underset{\sim}{R}$ 的隶属度函数 $\underset{\sim}{R}(x, y)$ 表示了 X 中的元素 x 与 Y 中的元素 y 具有这种关系的程度。X 到 X 的模糊关系称为 X 上的模糊关系。

由定义可见，模糊关系和模糊集合一样，完全由其隶属度函数 $\underset{\sim}{R}(x, y)$ 来刻划。当 $\underset{\sim}{R}(x, y)$ 仅取 1 或 0 两个值时，$\underset{\sim}{R}$ 就退化为经典集合，模糊关系退化为经典关系。因此可以认为，经典关系是模糊关系的特例。

类似 n 元关系的定义，可以得到如下的 n 元模糊关系定义。

定义 4：n 元模糊关系　设 X_1，X_2，…X_n 是 n 个集合，直积空间

$$X_1 \times X_2 \times \ldots \times X_n = \{(x_1, x_2, \ldots, x_n) \mid x_i \in X_i, i = 1,2,\ldots,n\}$$

上的一个 n 元模糊关系 $\underset{\sim}{R}$ 是指 $X_1 \times X_2 \times \ldots \times X_n$ 上的一个模糊子集，$\underset{\sim}{R}$ 由其隶属度函数 $\underset{\sim}{R}(x_1, x_2, \ldots, x_n)$ 来描述，它反映了 $(x_1, x_2, \ldots x_n)$ 具有这种关系的程度。

5.2 模糊推理系统的基本构成与建立步骤

5.2.1 模糊推理系统的基本类型

模糊推理系统可以分为三类，即纯模糊逻辑系统、具有模糊产生器和模糊消除器的模糊逻辑系统以及高木-关野（Takagi-Sugeno）型模糊逻辑系统。

1 纯模糊逻辑系统

纯模糊逻辑系统的输入与输出均为模糊集合，其框图如图 5.2.1。图中的模糊规则库由若干“如果-则”规则构成。模糊推理机在模糊推理系统中起着核心作用，它将输入模糊集合按照模糊规则映射为输出模糊集合。纯模糊逻辑系统的输入与输出均为模糊集合，它提供了一种量化专家语言信息和在模糊逻辑原则下系统地利用这类语言信息的一般化模式。

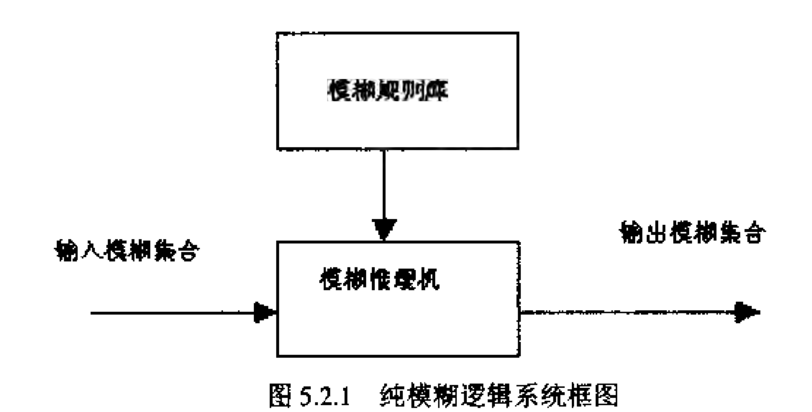


图 5.2.1 纯模糊逻辑系统框图

由于纯模糊逻辑系统的输入和输出均为模糊集合，而现实世界大多数工程系统的输入与输出都是精确值，因此纯模糊逻辑系统不能直接用于实际工程应用。为解决这一问题，有关学者在纯模糊逻辑系统的基础上提出了具有模糊产生器和模糊消除器的模糊逻辑系统，而日本学者高木(Takagi)和关野(Sugeno)则提出了模糊规则的后件为精确值的模糊逻辑系统，称为高木-关野型模糊逻辑系统。

2 具有模糊产生器和模糊消除器的模糊逻辑系统

在纯模糊逻辑系统的输入和输出部分分别添加模糊产生器和模糊消除器，得到的模糊逻辑系统的输入与输出均为精确量，因而可以直接在实际工程中加以应用。这类模糊逻辑系统就称为具有模糊产生器和模糊消除器的模糊逻辑系统。由于其应用的广泛性，通常就将其简称为模糊逻辑系统。这类模糊逻辑系统的结构如图 5.3.2 所示。

3 高木-关野型模糊逻辑系统

高木-关野型模糊逻辑系统是一类较为特殊的模糊逻辑系统，其模糊规则不同于一般的

模糊规则形式。通常的模糊规则的前件和后件均为模糊语言值，即具有如下形式：

$$\text{IF}\quad x_1 \text{ 是 } A_1, x_2 \text{ 是 } A_2, \ldots x_n \text{ 是 } A_n, \text{ THEN}\quad y \text{ 是 } B$$

其中，A_i（i=1,2,...n）是输入模糊语言值，B 是输出模糊语言值。

高木－关野模糊逻辑系统中，采用如下形式的模糊规则：

$$\text{IF}\quad x_1 \text{ 是 } A_1, x_2 \text{ 是 } A_2, \ldots x_n \text{ 是 } A_n, \text{ THEN}\quad y=\sum_{i=1}^{n} c_i x_i$$

其中，A_i（i=1,2,...n）是输入模糊语言值，c_i（i=1,2,...n）是真值参数。可以看出，高木－关野模糊逻辑系统的输出量在没有模糊消除器的情况下仍然是精确值。这类模糊逻辑系统的优点是输出量可用输入值的线性组合来表示，因而能够利用参数估计方法来确定系统的参数 c_i（i=1,2,...n）；同时，可以应用线性控制系统的分析方法来近似分析和设计模糊逻辑系统。其缺点是规则的输出部分不具有模糊语言值的形式，因此不能充分利用专家的控制知识，模糊逻辑的各种不同原则在这种模糊逻辑系统中应用的自由度也受到限制。

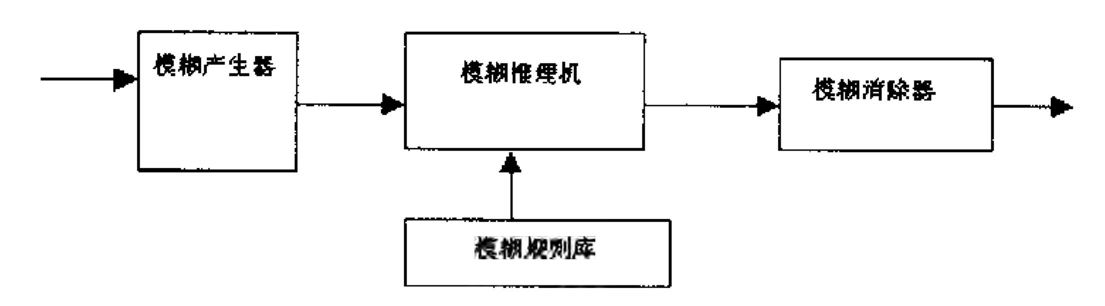


图 5.2.2　具有模糊产生器和模糊消除器的模糊逻辑系统

5.2.2　模糊逻辑系统的构成与主要设计步骤

前面讨论了模糊逻辑系统的基本类型，其中的具有模糊产生器和模糊消除器的模糊逻辑系统应用最为广泛。在 Matlab 模糊逻辑工具箱中主要针对这一类型的模糊逻辑系统提供了分析和设计手段，但同时对高木－关野模糊逻辑系统也提供了一些相关函数。下面将以具有模糊产生器和模糊消除器的模糊逻辑系统（以下简称为模糊逻辑系统）作为主要讨论对象。

构造一个模糊逻辑系统，首先必须明确其主要组成部分。一个典型的模糊逻辑系统主要由如下几个部分组成：

(1) 输入与输出语言变量，包括语言值及其隶属度函数；

(2) 模糊规则；

(3) 输入量的模糊化方法和输出变量的去模糊化方法；

(4) 模糊推理算法。

针对模糊逻辑系统的以上主要构成，在 Matlab 模糊逻辑工具箱中构造一个模糊推理系统有如下步骤：

(1) 建立模糊推理系统对应的数据文件，其后缀为.fis，用于对该模糊系统进行存储、修改和管理；

(2) 确定输入、输出语言变量及其语言值；

(3) 确定各语言值的隶属度函数，包括隶属度函数的类型与参数；
(4) 确定模糊规则；
(5) 确定各种模糊运算方法，包括模糊推理方法、模糊化方法、去模糊化方法等。

5.3 利用模糊逻辑工具箱建立模糊推理系统

5.3.1 模糊推理系统的建立、修改与存储管理

前面讨论了模糊推理系统的主要构成部分，即一个模糊推理系统由输入、输出语言变量及其隶属度函数、模糊规则、模糊推理机和去模糊化方法等各部分组成，在 Matlab 模糊逻辑工具箱中，把模糊推理系统的各部分作为一个整体，并以文件形式对模糊推理系统进行建立、修改和存储等管理功能。表 5.3.1 所示为该工具箱提供的有关模糊推理系统管理的函数及其功能。

表 5.3.1 模糊推理系统的管理函数

函数名称	功能
newfis	创建新的模糊推理系统
readfis	从磁盘读出存储的模糊推理系统
getfis	获得模糊推理系统的特性数据
writefis	保存模糊推理系统
showfis	显示添加注释了的模糊推理系统
setfis	设置模糊推理系统的特性
plotfis	图形显示模糊推理系统的输入-输出特性

1 newfis

功能：创建新的模糊推理系统。

格式：a=newfis(fisName,fisType,andMethod,orMethod,impMethod, aggMethod, defuzzMethod)

说明：该函数用于创建一个新的模糊推理系统，模糊推理系统的特性可由函数的参数指定，其参数个数可达 7 个。

参数的含义说明如下：

fisName——模糊推理系统名称；
fisType——模糊推理类型；
andMethod——与运算操作符；
orMethod——或运算操作符；
impMethod——模糊蕴含方法；
aggMethod——各条规则推理结果的综合方法；
defuzzMethod——去模糊化方法。

举例:

```
a=newfis('newsys');
getfis(a)
```

输出结果:

```
Name = newsys
Type = mamdani
NumInputs = 0
InLabels =
NumOutputs = 0
OutLabels =
NumRules 0
AndMethod min
OrMethod max
ImpMethod min
AggMethod max
DefuzzMethod centroid
```

2　readfis

功能: 从磁盘中加载模糊推理系统。

格式: readfis('filename')

说明: 打开一个由 filename 指定的模糊推理系统的数据文件(.fis)并将其加载到当前的工作空间(Workspace)中。当未指定文件名时，Matlab 将会打开一个文件对话窗口，提示用户指定某一.fis 文件。

举例:

```
fismat = readfis('tipper');
getfis(fismat)
```

输出结果:

```
Name = tipper
Type = mamdani
NumInputs = 2
InLabels =
service
food
NumOutputs = 1
OutLabels =
tip
NumRules = 3
AndMethod = min
OrMethod = max
```

```
ImpMethod = min
AggMethod = max
DefuzzMethod = centroid
```

3 getfis

功能：获得模糊推理系统的属性。

格式：getfis(a)

getfis(a,'fisprop')

getfis(a,'vartype',varindex,'varprop')

getfis(a,'vartype',varindex,'mf',mfindex)

getfis(a,'vartype',varindex,'mf',mfindex,'mfprop')

说明：使用该函数是获得模糊推理系统及其对应矩阵的所有属性的基本方法。在参数列表中，a 为模糊推理系统在内存中对应的矩阵，且是必须指定的参数；后面的其他参数则可以省略。当仅有参数 a 时，函数返回模糊推理系统的所有属性。

举例：

```
a = readfis('tipper');
getfis(a)
```

输出：

```
Name = tipper
Type = mamdani
NumInputs = 2
InLabels =
Service
food
NumOutputs = 1
OutLabels =
tip
NumRules = 3
AndMethod = min
OrMethod = max
ImpMethod = min
AggMethod = max
DefuzzMethod = centroid
```

参数 fisprop 用于指定期望获得的某一属性。此时 getfis 的输出结果仅包括指定的属性值：

```
getfis(a,'type')
```

输出为：

```
        mamdani
```

当属性参数为 input 或 output 时，后面的第 3 个参数用于指定某一个输入或输出语言变量：

```
getfis(a,'input',1)
Name = service
NumMFs = 3
MFLabels =
poor
good
excellent
            Range = [0 10]
```

上例中的参数 input 用于指定返回某一输入语言变量，其后的参数 1 指定返回第一个输入语言变量的有关属性，包括名称、语言值的个数和名称以及该语言变量的论域范围。函数 getfis 可以进一步增加参数以返回指定的输入或输出语言变量的某一属性值。如在第 4 个参数中指定返回语言变量的名称

```
getfis(a,'input',1, name')
```

输出结果：

```
service
```

getfis 函数在某些情况下还可以有更多的参数，例如下面的两个例子：

```
getfis(a,'input',1,'mf',2)
```

输出结果：

```
Name = good
Type = gaussmf
Params =
1.5000 5.0000
getfis(a,'input',1,'mf',2,'name')
```

输出结果：

```
good
```

在第一个例子中，参数 'mf' 和 2 指定返回输入语言变量的第 2 个语言值的隶属度函数属性，包括类型和参数。在第二个例子中，参数 'name' 指定返回第 2 个语言值的名称。

4 writefis

功能： 将模糊推理系统以矩阵形式保存在内存中的数据写入磁盘文件中。

格式： writefis(fismat)

writefis(fismat,filename)

writefis(fismat,filename,'dialog')

说明： 模糊推理系统在内存中的数据是以矩阵形式存储的，其对应的矩阵名为 fismat。当需要将模糊推理系统的数据写入磁盘文件时，就可以利用 writefis 函数。在只有一个参数即 writefis(fismat)的情况下，Matlab 将

打开一个文件对话窗口，提示用户选择某一磁盘文件或输入一个新的文件名；用户也可直接在函数的第 2 个参数 filename 中指定某一磁盘文件名：writefis(fismat,filename,'dialog')则打开一个以 filename 为缺省文件名的对话窗口，用户仍可重新输入文件名。文件名的后缀缺省为.fis。

举例:

```
a = newfis('tipper');
a = addvar(a,'input','service',[0 10]);
a = addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a = addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a = addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
writefis(a,'my_file')
```

5 showfis

功能: 以分行的形式显示模糊推理系统矩阵的所有属性。

格式: showfis(fismat)

说明: fismat 为模糊推理系统在内存中的矩阵表示。

举例:

```
a = readfis('tipper');
showfis(a)
```

6 setfis

功能: 设置模糊推理系统的属性。

格式: a = setfis(a,'propname',newprop)

a = setfis(a,'vartype',varindex,'propname',newprop)

a = setfis(a,'vartype',varindex,'mf',mfindex, ...

propname',newprop);

说明: 该函数的参数个数可以有 3、5、7 三种情况。当参数个数为 3 时，用于设定模糊推理系统的全局属性，包括：

- name——模糊推理系统的名称；
- type——模糊推理系统的类型；
- numinputs——模糊推理系统的输入变量个数；
- numoutputs——模糊推理系统的输出变量个数；
- numrules——规则个数；
- andmethod——与运算方法；
- ormethod——或运算方法；
- impmethod——模糊蕴含方法；
- aggmethod——各个规则推理结果的综合方法；
- defuzzmethod——输出去模糊化方法。

当参数个数为5个时，用于设定模糊推理系统矩阵某一个语言变量的属性，这些属性包括：name（变量名称），bounds（论域范围）。

当参数个数为7时，用于设定一个语言变量的某一隶属度函数的属性，包括：

name（隶属度函数名称），type（类型），params（参数）。

举例:

例1. 三个参数的情况

```
a = readfis('tipper');
a2 = setfis(a,'numinputs',3);
getfis(a2,'numinputs')
```

输出结果

```
3
```

例2. 五个参数的情况

```
a2 = setfis(a,'input',1,'name','help');
getfis(a2,'input',1,'name')
```

输出结果

```
help
```

例3. 七个参数的情况

```
a2 = setfis(a,'input',1,'mf',2,'name','wretched');
getfis(a2,'input',1,'mf',2,'name')
```

输出结果

```
wretched
```

5.3.2 输入输出语言变量及其语言值

在模糊推理系统中，专家的控制知识以模糊规则形式表示。为直接反映人类自然语言的模糊性特点，模糊规则的前件和后件中引入语言变量和语言值的概念。语言变量分为输入语言变量和输出语言变量，输入语言变量是对模糊推理系统输入变量的模糊化描述，通常位于模糊规则的前件中，输出语言变量是对模糊推理系统输出变量的模糊化描述，通常位于模糊规则的后件中。语言变量具有多个语言值，每个语言值对应一个隶属度函数。语言变量的语言值构成了对输入和输出空间的模糊分割，模糊分割的个数即语言值的个数以及语言值对应的隶属度函数决定了模糊分割的精细化程度。模糊分割的个数也决定了模糊规则的个数，模糊分割数越多，控制规则数也越多。因此在设计模糊推理系统时，应在模糊分割的精细程度与控制规则的复杂性之间取得折衷。

在 Matlab 模糊逻辑工具箱中提供了向模糊推理系统添加或删除语言变量及其语言值的函数，即 advar 和 rmvar。

1 advar

功能：向模糊推理系统添加语言变量。

格式：a = addvar(a,varType,varName,varBounds)

说明：在参数列表中，a为模糊推理系统的对应矩阵名称，varType用于指定语言变量的类型；varName用于指定语言变量的名称；varBounds用于指定变量的论域范围。对于添加到同一个模糊推理系统的语言变量，将按照添加的先

后顺序分别赋予一个编号，编号从1开始，逐渐递增。对输入与输出语言变量则独立地分开编号。

举例：
```
a=newfis('tipper');
a=addvar(a,'input','service',[0 10]);
getfis(a,'input',1)
```
输出结果：
```
Name = service
NumMFs = 0
MFLabels =
Range = [0 10]
```

2 rmvar

功能：从模糊推理系统中删除语言变量。

格式：rmvar(a,'varType',varIndex)。

说明：当一个模糊语言变量正在被当前的模糊规则集使用时，则不能删除该变量。在一个模糊语言变量被删除后，Matlab模糊逻辑工具箱将会自动地对模糊规则集进行修改，以保证一致性。

举例：

例1
```
a = newfis('mysys');
a = addvar(a,'input','temperature',[0 100]);
getfis(a)
```
输出结果：
```
Name = mysys
Type = mamdani
NumInputs = 1
InLabels =
temperature
NumOutputs = 0
OutLabels =
NumRules = 0
```

例2
```
b = rmvar(a,'input',1);
getfis(b)
```
输出结果：
```
Name = mysys
Type = mamdani
NumInputs = 0
InLabels =
NumOutputs = 0
```

```
OutLabels =
NumRules = 0
```

5.3.3 模糊语言变量的隶属度函数

每个模糊语言变量具有多个模糊语言值。模糊语言值的名称通常具有一定的含义，如 NB（负大）、NM（负中）、NS（负小）、ZE（零）、PS（正小）、PM（正中）、PB（正大）等。每个语言值都对应一个隶属度函数。隶属度函数可有两种描述方式，即数值描述方式和函数描述方式。数值描述方式适用于语言变量的论域为离散的情形，此时隶属度函数可用向量或表格的形式来表示；对于论域为连续的情况，隶属度函数则采用函数描述方式。表 5.3.2 所示即为一个采用数值方法描述的模糊语言变量隶属度函数。

表 5.3.2 数值方法描述的隶属度函数

元素 / μ / 模糊集	-4	-3	-2	-1	0	1	2	3	4
NB	0.8	0.6	0.2	0.1	0	0	0	0	0
NS	0.6	0.8	1.0	0.6	0.2	0.1	0	0	0
ZE	0	0.2	0.6	0.8	1.0	0.6	0.2	0.1	0
PS	0	0	0	0.2	0.6	0.8	1.0	0.6	0.2
PB	0	0	0	0	0	0.1	0.2	0.6	0.8

采用函数描述的隶属度函数往往具有多种形式，常见的有高斯型、三角型、梯形、S型等。隶属度函数曲线的形状决定了对输入、输出空间的模糊分割，对模糊推理系统的性能有重要的影响。在Matlab模糊逻辑工具箱中提供了丰富的隶属度函数类型的支持，利用工具箱的有关函数可以方便地对各类隶属度函数进行建立、修改和删除等操作。下面首先对模糊逻辑工具箱中有关模糊推理系统中隶属度函数操作的功能进行介绍。

1 addmf

功能: 向模糊推理系统的语言变量添加隶属度函数。

格式: a = addmf(a,varType,varIndex,mfName,mfType,mfParams)

说明: 隶属度函数只能作为模糊推理系统中存在的某一语言变量的语言值添加，而不能添加到一个不存在的语言变量中。某个语言变量的隶属度函数（即语言值）按照添加的顺序加以编号，第一个添加的隶属度函数被编为 1 号，此后依次递增。在参数列表中，a 为模糊推理系统的对应矩阵，varType 指定语言变量的类型（输入或输出），varIndex 指定语言变量的编号，mfName 指定隶属度函数的名称，mfType 和 mfParams 分别指定隶属度函数的类型和参数。

举例:

```
a=newfis('tipper');
a=addvar(a,'input','service',[0 10]);
a=addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a=addmf(a,'input',1,'good','gaussmf',[1.5 5]);
```

```
a=addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
plotmf(a,'input',1)
```

隶属度函数曲线如图5.3.1所示。

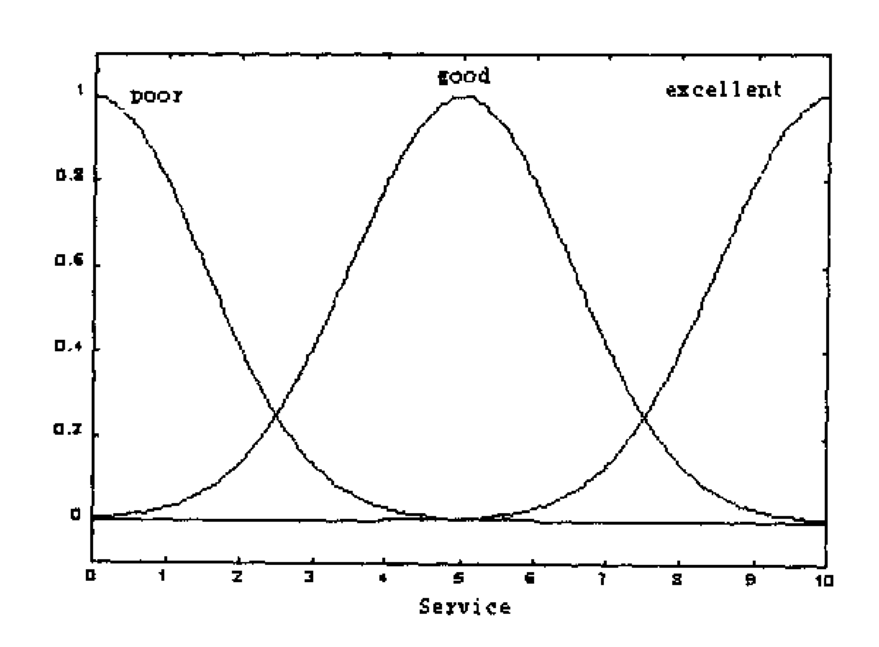


图5.3.1 隶属度函数曲线

2 plotmf

功能： 绘制语言变量所有语言值的隶属度函数曲线。

格式： plotmf(fismat,varType,varIndex)

说明： 在参数列表中，各参数的含义如下：

fismat——模糊推理系统的对应矩阵名称；

varType——语言变量的类型；

varIndex——语言变量的编号。

3 rmmf

功能： 从模糊推理系统中删除一个语言变量的某一隶属度函数。

格式： a = rmmf(a,'varType',varIndex,'mf',mfIndex)

说明： 当一个隶属度函数正在被当前模糊推理规则使用时，则不能删除。各参数的含义说明如下：

varType——语言变量的类型；

varIndex——语言变量的编号；

mf——隶属度函数的名称；

mfIndex——隶属度函数的编号。

举例：

```
a = newfis('mysys');
a = addvar(a,'input','temperature',[0 100]);
```

```
a = addmf(a,'input',1,'cold', trimf',[0 30 60]);
getfis(a,'input',1)
Name = temperature
NumMFs = 1
MFLabels = cold
Range = [0 100]
b = rmmf(a,'input',1,'cold',1);
getfis(b,'input',1)
Name = temperature
NumMFs = 0
MFLabels =
Range = [0 100]
```

在Matlab模糊逻辑工具箱中支持的隶属度函数类型有如下几种：高斯型、三角型、梯形、钟型、Sigmoid型、π 型以及Z型。利用工具箱中提供的函数可以建立和计算上述各种类型隶属度函数。

4　gaussmf

功能： 建立高斯型隶属度函数。

格式： y = gaussmf (x,params)

y = gaussmf (x,[sig c])

说明： 高斯型函数的形状由两个参数决定：sig和c，其中c决定了函数的中心点，sig决定了函数曲线的宽度σ。高斯函数的表达式如下：

$$y = e^{-\frac{(x-c)^2}{\sigma^2}}$$

参数x用于指定变量的论域。

举例： 建立高斯型隶属度函数，如图5.3.2所示。

```
x=0:0.1:10;
y=gaussmf(x,[2 5]);
plot(x,y)
xlabel('gaussmf, P=[2 5]')
```

5　gauss2mf

功能： 建立双边高斯型隶属度函数。

格式： y = gauss2mf (x,params)

y = gauss2mf (x,[sig1 c1 sig2 c2])

说明： 双边高斯型函数的曲线由两个中心点相同的高斯型函数的左、右半边曲线组合而成，其表达式如式5.3.1 所示。

参数sig1 、c1、 sig2、 c2分别对应左、右半边高斯函数的宽度与中心点，c1<c2。

举例：建立双边高斯型隶属度函数，如图5.3.3所示。

```
x=0:0.1:10;
y=gauss2mf(x,[1 3 3 4]);
plot(x,y)
xlabel('gauss2mf, P=[1 3 3 4]')
```

$$y=\begin{cases} e^{-\frac{(x-c)^2}{\sigma_1^2}}, & x<c \\ e^{-\frac{(x-c)^2}{\sigma_2^2}}, & x\geq c \end{cases} \tag{5.3.1}$$

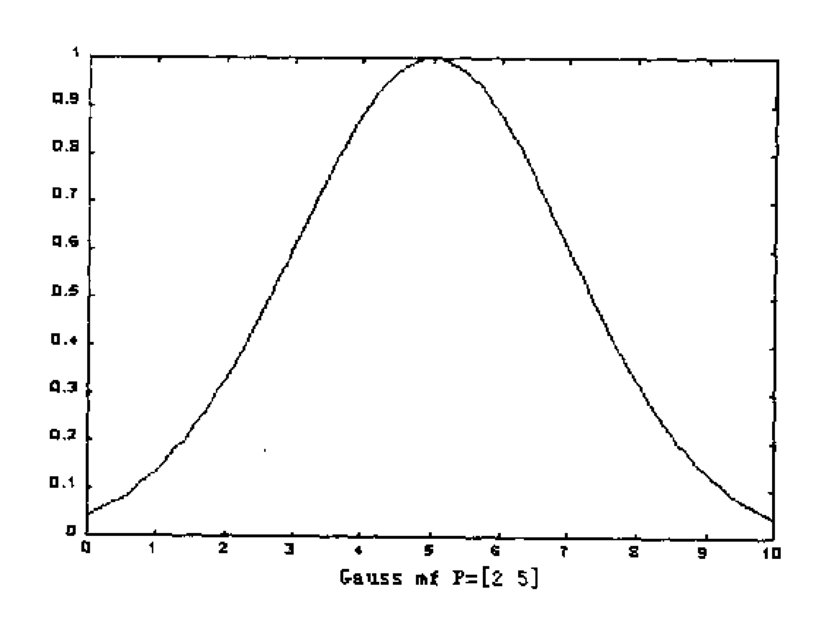


图5.3.2 高斯型隶属度函数曲线

6 gbellmf

功能：建立一般的钟形隶属度函数。

格式：y = gbellmf(x,params)

y = gbellmf(x,[a b c])

说明：参数x指定变量的论域范围，[a b c]指定钟形函数的形状，钟形函数的表达式如下：

$$y=\frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}}$$

举例：建立并绘制钟形隶属度函数曲线，如图 5.3.4 所示。

```
x=0:0.1:10;
y=gbellmf(x,[2 4 6]);
plot(x,y)
```

```
xlabel('gbellmf, =[2 4 6]')
```

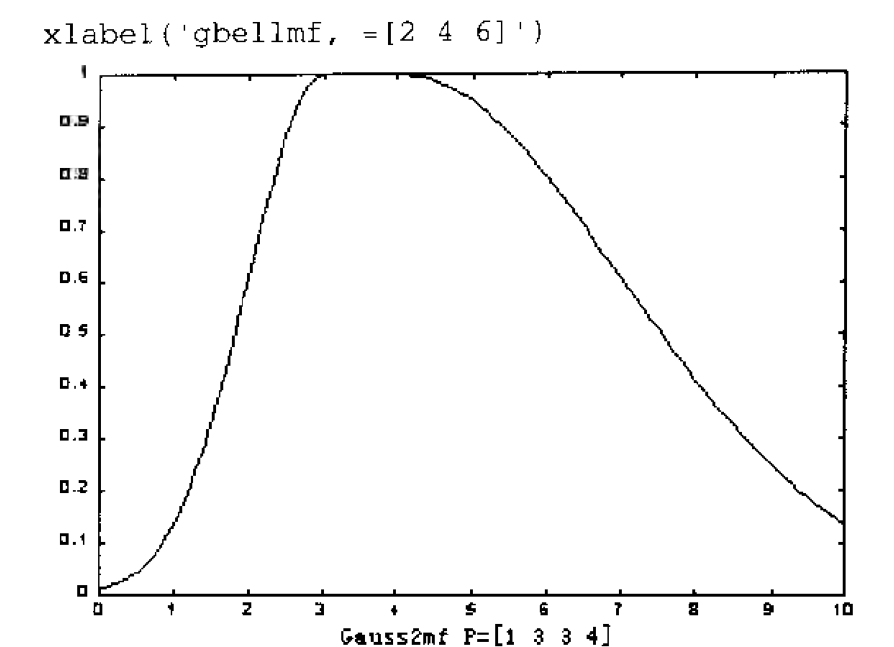


图5.3.3　双边高斯型隶属度函数曲线

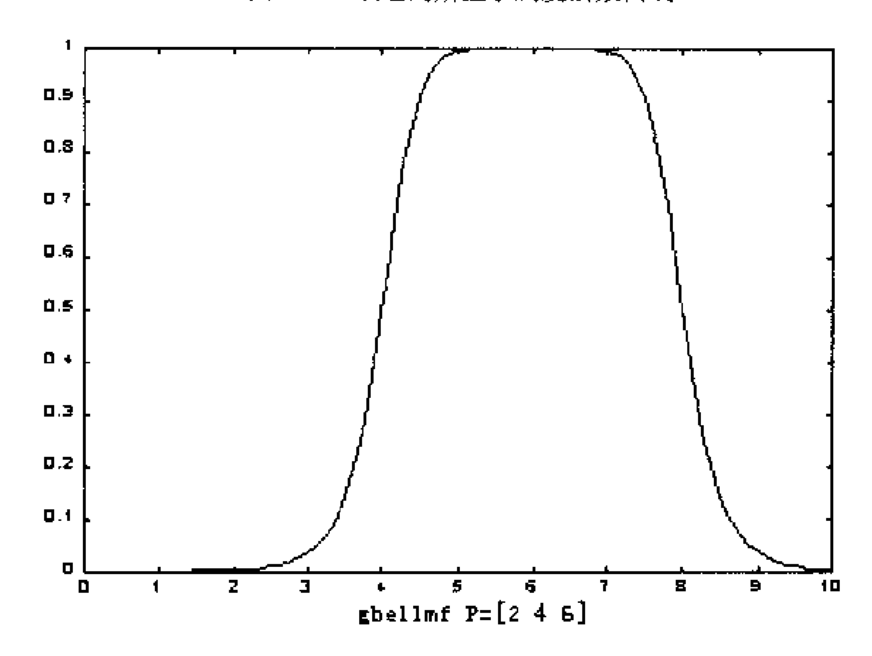


图5.3.4　钟型隶属度函数曲线

7　pimf

功能：建立π 型隶属度函数。

格式：y = pimf(x,params)

y = pimf(x,[a b c d])

说明：π 型函数是一种基于样条的函数，由于其形状类似字母π 而得名。参数x指定函数的自变量范围，[a b c d]决定函数的形状，在图5.3.5中，a,b分别对应曲线下部的左右两个拐点，b和c分别对应曲线上部的左右两个拐点。

举例:

```
x=0:0.1:10;
y=pimf(x,[1 4 5 10]);
plot(x,y)
xlabel('pimf, P=[1 4 5 10]')
```

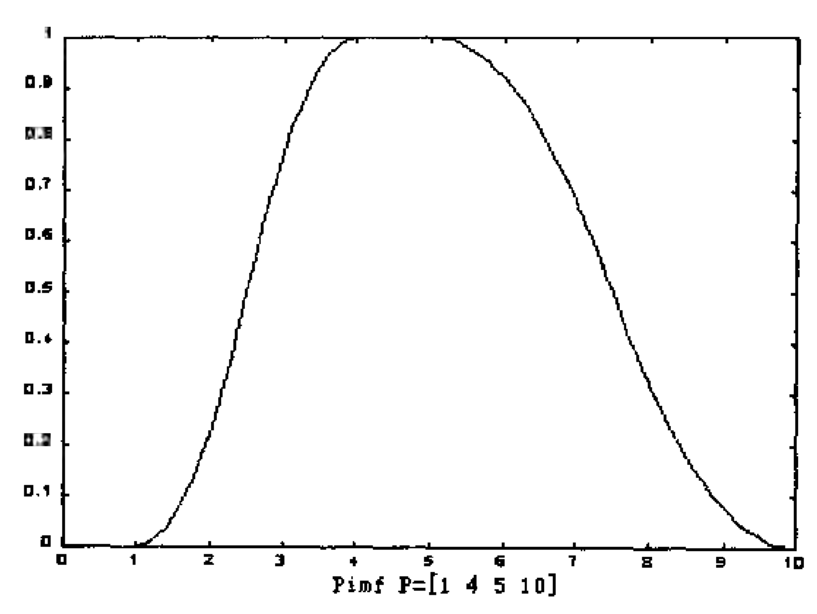


图5.3.5 π 型隶属度函数曲线

8 sigmf

功能：建立sigmoid型隶属度函数。

格式：y = sigmf(x,params)

y = sigmf(x,[a c])

说明：参数x用于指定变量的论域范围，[a c]决定了sigmoid型函数的形状，其表达式如下：

$$y = \frac{1}{1 + e^{-a(x-c)}}$$

sigmoid型函数曲线具有半开的形状，因而适于作为“极大”、“极小”等语言值的隶属度函数。为了得到更符合人们习惯的隶属度函数形状，可以利用两个sigmoid型函数之和或乘积来构造新的隶属度函数类型，模糊逻辑工具箱中提供了相应的函数，参见dsigmf和psigmf。

举例：建立sigmoid型隶属度函数，如图5.3.6所示。

```
x=0:0.1:10;
y=sigmf(x,[2 4]);
plot(x,y)
xlabel('sigmf, P=[2 4]')
```

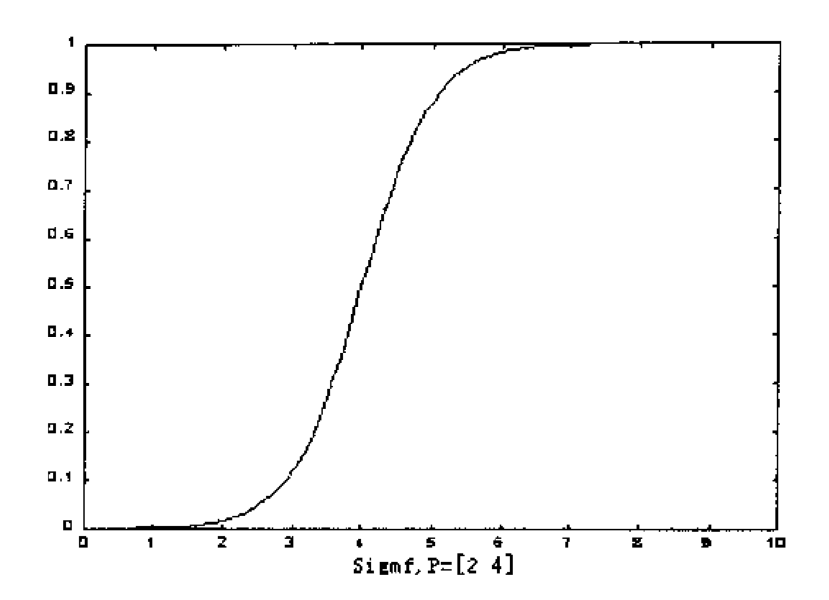


图5.3.6　sigmoid型隶属度函数曲线

9 psigmf

功能：通过两个sigmoid型函数的乘积来构造新的隶属度函数。

格式：y = psigmf(x,params)

　　　y = psigmf(x,[a1 c1 a2 c2])

说明：参数a1、c1和a2、c2分别用于指定两个sigmoid型函数的形状，参数x指定变量的利用范围。新的函数表达式如下：

$$y=\frac{1}{(1+e^{-a_1(x-c_1)})(1+e^{-a_2(x-c_2)})}$$

举例：由两个sigmoid型函数的乘积来构造新的隶属度函数，如图5.3.7所示。

```
x=0:0.1:10;
y=psigmf(x,[2 3 -5 8]);
plot(x,y)
xlabel('psigmf, P-[2 3  5 8]')
```

10 dsigmf

功能：通过计算两个 sigmoid 型函数之和来构造新的隶属度函数。

格式：y = dsigmf(x,params)

　　　y = dsigmf(x,[a1 c1 a2 c2])

说明：本函数的用法与psigmf类似，参数a1、c1和a2、c2分别用于指定两个sigmoid型函数的形状，构造得到的新的隶属度函数表达式为：

$$y=\frac{1}{1+e^{-a_1(x-c_1)}}+\frac{1}{1+e^{-a_2(x-c_2)}}$$

举例：绘制两个sigmoid型函数之和的隶属度函数曲线，如图5.3.8所示。

```
x=0:0.1:10;
y=dsigmf(x,[5 2 5 7]);
plot(x,y)
xlabel('dsigmf, P=[5 2 5 7]')
```

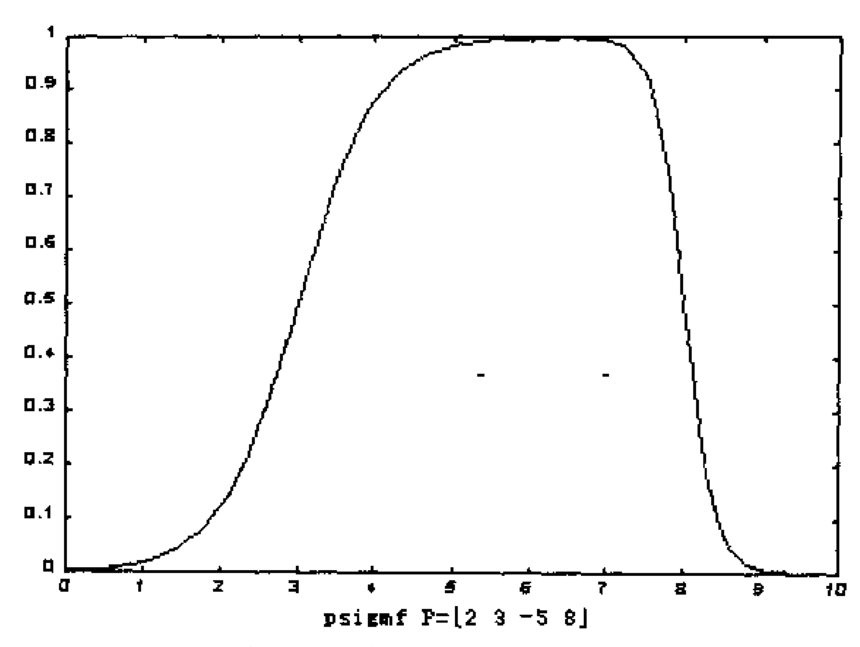


图5.3.7　两个sigmoid型函数的乘积

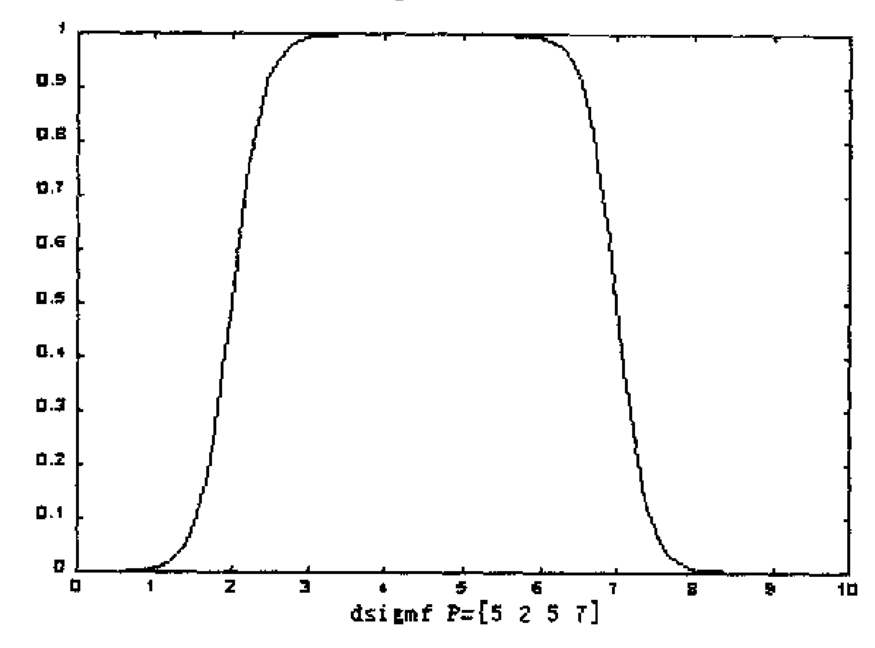


图5.3.8　两个sigmoid型函数之和

11　trapmf

功能：建立梯形隶属度函数。

格式：y = trapmf(x,params)

　　　y = trapmf(x,[a b c d])

说明：参数x指定变量的论域范围，参数a、b、c和d指定梯形隶属度函数的形状，其对应的表达式如下：

$$f(x,a,b,c,d)=\begin{cases}0, & x\le a\\ \dfrac{x-a}{b-a}, & a\le x\le b\\ 1, & b\le x\le c\\ \dfrac{d-x}{d-c}, & c\le x\le d\\ 0, & d\le x\end{cases}$$

举例：建立并绘制梯形隶属度函数曲线，如图5.3.9所示。

```
x=0:0.1:10;
y=trapmf(x,[1 5 7 8]);
plot(x,y)
xlabel('trapmf, P=[1 5 7 8]')
```

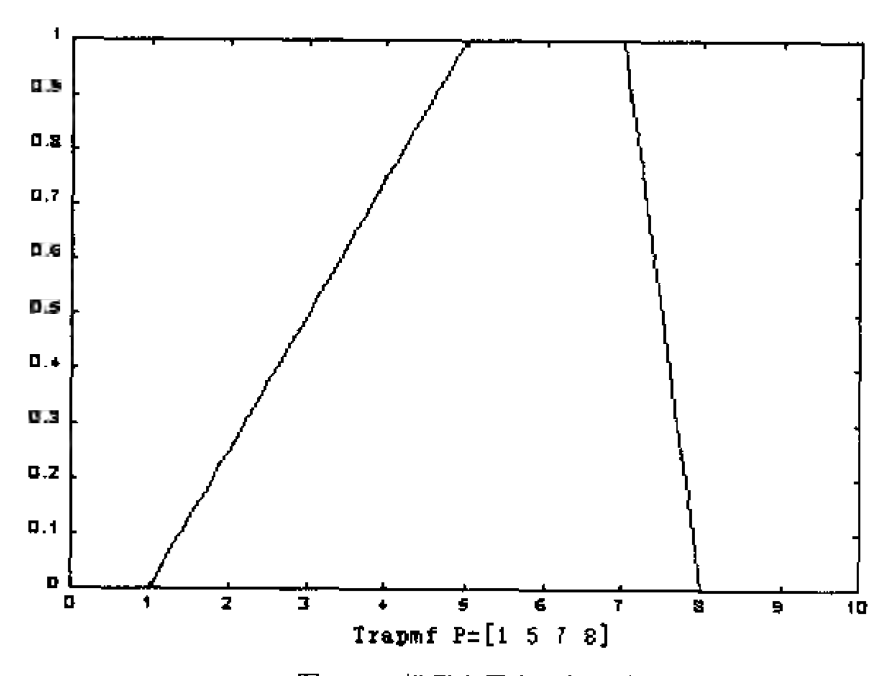


图5.3.9　梯形隶属度函数曲线

12　trimf

功能：建立三角形隶属度函数。

格式：y = trimf(x,params)

y = trimf(x,[a b c])

说明：参数x指定变量的论域范围，参数a、b和c指定三角形函数的形状，其表达式如下：

$$f(x,a,b,c,d)=\begin{cases}0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & c \le x \le d \\ 0, & c \le x\end{cases}$$

举例：建立三角形隶属度函数并绘制曲线，如图5.3.10所示。

```
x=0:0.1:10;
y=trimf(x,[3 6 8]);
plot(x,y)
xlabel('trimf, P=[3 6 8]')
```

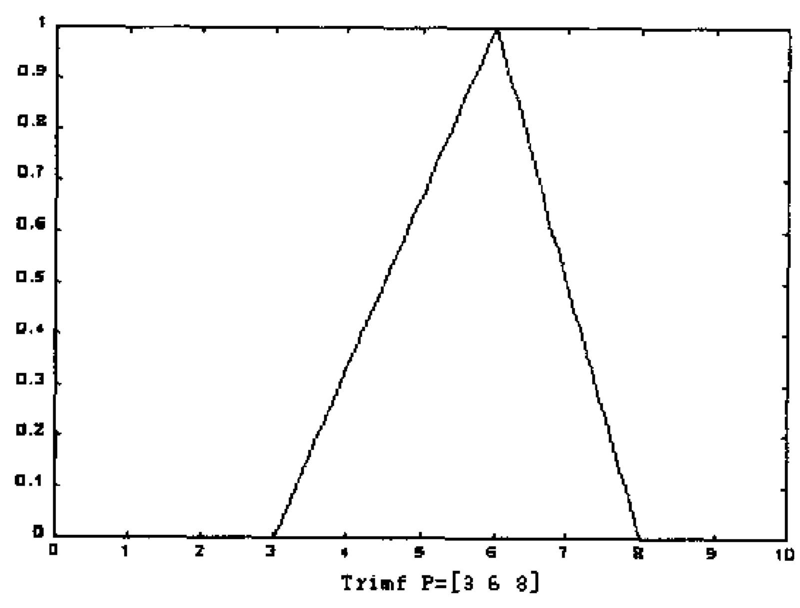


图5.3.10　三角形隶属度函数

13　zmf

功能：建立Z形隶属度函数曲线。

格式：y = zmf (x,params)

　　　y = zmf (x,[a b])

说明：Z形函数是一种基于样条插值的函数，两个参数a和b分别定义样条插值的起点和终点；参数x指定变量的论域范围。

举例：建立Z形隶属度函数并绘制曲线，如图5.3.11所示。

```
x=0:0.1:10;
y=zmf(x,[3 7]);
plot(x,y)
xlabel('zmf, P=[3 7]')
```

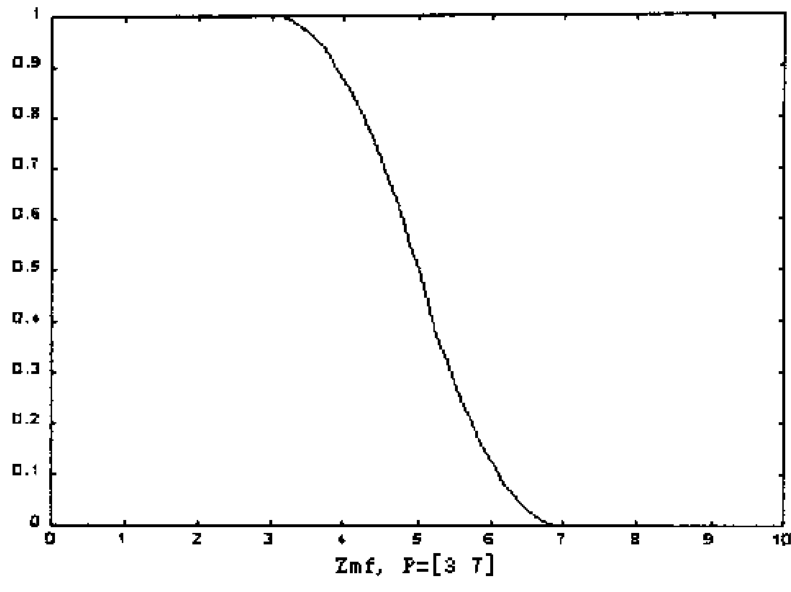


图5.3.11　Z形隶属度函数曲线

在Matlab模糊逻辑工具箱中还提供不同类型隶属度函数之间参数转换的功能，其对应的函数为mf2mf。

14　mf2mf

功能：进行不同类型隶属度函数之间的参数转换。

格式：outParams = mf2mf(inParams,inType,outType)

说明：该函数将尽量保持两种类型的隶属度函数曲线在形状上的近似，特别是保持隶属度等于0.5处的点的重合。但不可避免地会丢失一些信息，所以当再次使用该函数进行反向转换时将无法得到与原来函数相同的参数。

举例：实现钟型隶属度函数向三角形隶属度函数的转换，如图5.3.12所示。

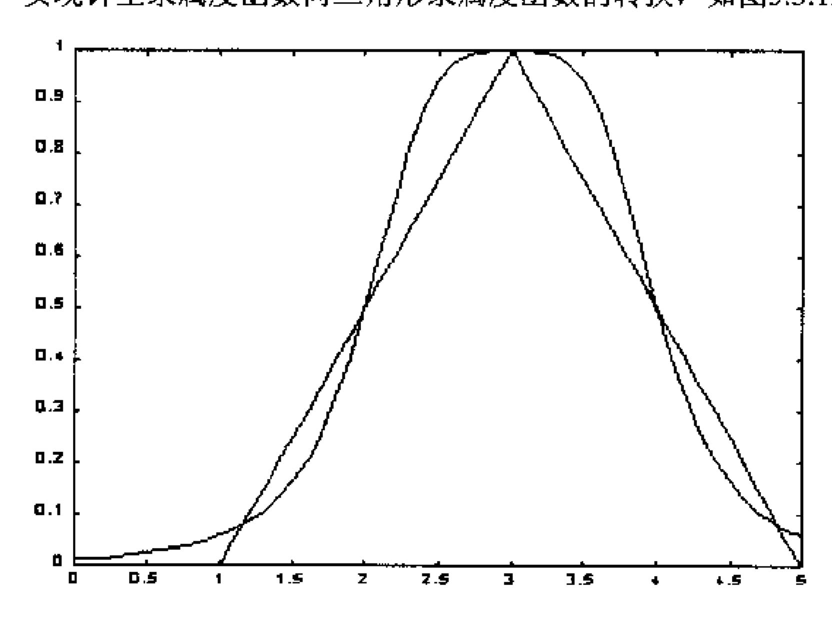


图 5.3.12　隶属度函数参数的转换

```
x=0:0.1:5;
mfp1 = [1 2 3];
mfp2 = mf2mf(mfp1,'gbellmf','trimf');
plot(x,gbellmf(x,mfp1),x,trimf(x,mfp2))
```

以上对Matlab模糊逻辑工具箱中与语言变量及其隶属度函数有关的函数进行了详细的介绍，表5.3.2 列出了上述所有的函数。

表 5.3.2 与语言变量和隶属度相关的功能函数

函数名称	功能
addvar	添加模糊语言变量
addmf	添加模糊语言变量的隶属度函数
dsigmf	计算两个 Sigmoid 型隶属度函数之差
gauss2mf	建立双边高斯型隶属度函数
gaussmf	建立高斯型隶属度函数
mf2mf	隶属度函数间的参数转换
gbellmf	建立一般的钟型隶属度函数
plotmf	绘制隶属度函数曲线
pimf	建立 π 型隶属度函数
psigmf	计算两个 Sigmoid 型隶属度函数之积
rmmf	删除隶属度函数
rmvar	删除模糊语言变量
sigmf	建立 Sigmoid 型的隶属度函数
trapmf	建立梯形隶属度函数
trimf	建立三角形隶属度函数
zmf	建立 Z 型隶属度函数

5.3.4 模糊规则的建立与修改

在模糊推理系统中，模糊规则以模糊语言的形式描述人类的经验和知识，规则是否正确地反映人类专家的经验和知识，是否反映对象的特性，直接决定模糊推理系统的性能。通常模糊规则的形式是“IF 前件 THEN 后件”，前件由对模糊语言变量的语言值描述构成，如“温度较高，压力较低”；在一般的模糊推理系统中，后件由对输出模糊语言变量的语言值描述构成，但在高木–关野模糊推理系统中，后件将输出变量表示成输入量的精确值的组合。模糊规则的这种形式化表示是符合人们通过自然语言对许多知识的描述和记忆习惯的。

模糊规则的建立是构造模糊推理系统的关键，其建立方法主要有如下 3 种。

(1) 总结操作人员、专家的经验和知识：操作人员在长期从事仪器设备的操作中，积累了大量的经验，这些经验往往都具有模糊性的特点。总结这些经验对构造模糊规则有重要的指导意义；某个领域的专家则对该领域的各种过程机理有较深刻的认识，对过程的运行特性能够通过理论分析给出定性的结论以帮助建立模糊规则。

(2) 基于过程的模糊模型：被控过程的动态特性可以用模糊模型来描述，称为过程

的模糊模型。基于过程的模糊模型可以产生一组模糊控制规则来使被控过程到达期望的性能。这种方法存在的困难就是难于获得充分反映被控过程特性的模糊模型及其参数。

(3) 基于学习的方法。当被控过程存在时变的特性或难以直接构造模糊控制器时，可以通过设计具有自组织、自学习能力的模糊控制器来自动获得模糊规则。Procyk 和 Mamdani 首先提出了自组织模糊控制器的一种分级结构，包括两级控制规则，第一级直接用于控制对象，第二级根据测量数据和评价准则在线地修改第一级模糊规则。

上面介绍了建立模糊规则的三种主要方法，其中第一种方法是最基本的，也是应用最广泛的方法。在实际应用中，初步建立的模糊规则往往难以到达良好的效果，必须不断加以修正和试凑。在模糊规则的建立修正和试凑过程中，应尽量保证模糊规则的完备性和相容性。所谓模糊规则的完备性即对于控制过程的任一状态，模糊规则都能产生有关控制作用。而模糊规则的相容性则反映在输出模糊集合是否是多峰的，如果存在多峰的现象，则说明模糊规则中有相互矛盾的情况存在。

在 Matlab 模糊逻辑工具箱中提供了有关对模糊规则建立和操作的函数，下面给出详细说明。

1　addrule

功能：向模糊推理系统添加模糊规则。

格式：a = addrule(a,rulelist)

说明：参数a为模糊推理系统对应的矩阵名称，rulelist以向量的形式给出需要添加的模糊规则，该向量的格式有严格的要求，如果模糊推理系统有m个输入语言变量和n个输出语言变量，则向量rulelist的列数必须为m+n+2，而行数任意。在rulelist的每一行中，前m个数字表示各输入语言变量的语言值，其后的n个数字表示输出语言变量的语言值，第m+n+1个数字是该规则适用的权重，权重的值在0到1之间，一般设定为1；第m+n+2个数字为0或1两个值之一，如果为1则表示模糊规则前件的各语言变量之间是“与”的关系，如果是0则表示是“或”的关系。

举例：

```
ruleList=[1 1 1 1 11 2 2 1 1];
a = addrule(a,ruleList);
```

在上例中，第一条规则可以描述为：“如果输入1是隶属度函数1，输入2是隶属度函数1，那么输出1是隶属度函数1”。

2　parsrule

功能：解析模糊规则。

格式：fis2 = parsrule(fis,txtRuleList,ruleFormat)

说明：函数 parsrule 对给定的模糊语言规则进行解析并添加到模糊推理系统矩阵中，输入参数定义为：

fis——初始的模糊推理系统矩阵；

txtRuleList——模糊语言规则；

ruleFormat——规则的格式，包括语言型（'verbose'）、符号型（'symbolic'）和索引型（'indexed'）。

举例：

```
a = readfis('tipper');
ruleTxt = 'if service is poor then tip is generous';
a2 = parsrule(a,ruleTxt,'verbose');
showrule(a2)
```

输出为：

```
1. If (service is poor) then (tip is generous) (1)
```

3 showrule

功能：显示模糊规则。

格式：showrule(a,indexList,format)

说明：本函数用于显示指定的模糊推理系统的模糊规则，模糊规则可以按三种方式显示，即：详述方式（verbose）、符号方式（symbolic）和隶属度函数编号方式（membership function index referencing）。第一个参数是模糊推理系统矩阵的名称，第二个参数是规则编号，第三个参数是规则显示方式。规则编号可以以向量形式指定多个规则。

举例：

例1.
```
a = readfis('tipper');
showrule(a,1)
```

输出结果：

```
1. If (service is poor) or (food is rancid) then (tip is
cheap) (1)
```

例2.
```
showrule(a,2)
```

输出结果：

```
1.If (service is good) then (tip is average) (1)
```

例3.
```
showrule(a,[3 1],'symbolic')
```

输出结果：

```
3.(service==excellent) | (food==delicious) =>
                                   (tip=generous) (1)
1.(service==poor) | (food==rancid) => (tip=cheap) (1)
```

例4.
```
showrule(a,1:3,'indexed')
```

输出结果：

```
1 1, 1 (1) : 2
2 0, 2 (1) : 1
3 2, 3 (1) : 2
```

5.3.5 模糊推理计算与去模糊化

在建立输入输出语言变量及其隶属度函数，并构造完成模糊规则之后，就可执行模糊推理计算了。模糊推理的执行结果与模糊蕴含操作的定义、推理合成规则、模糊规则前件部分的连接词“AND”的操作定义等有关，因而有多种不同的算法。

目前常用的模糊推理合成规则是“极大–极小”合成规则，设 R 表示规则：“X 为 $A \to Y$ 为 B”表达的模糊关系，则当 X 为 A' 时，按照“极大–极小”规则进行模糊推理的结论 B' 计算如下：

$$B' = A' \circ R = \int_Y \bigvee_{x \in X} (\mu_{A'}(x) \wedge \mu_R(x, y)) / y$$

基于模糊蕴含操作的不同定义，人们提出了多种模糊推理算法，其中较为常用的是 Mamdani 模糊推理算法、Larsen 模糊推理算法。另外，对于输出为精确量的一类特殊模糊逻辑系统——Takagi–Sugeno 型模糊推理系统，采用了将模糊推理与去模糊化结合的运算操作。

1 常用的模糊推理算法

- **Mamdani 型模糊推理算法**

Mamdani 型模糊推理算法采用极小运算规则定义模糊蕴含表达的模糊关系，例如规则 R：IF x 为 A THEN y 为 B 表达的模糊关系 Rc 定义为：

$$\begin{aligned} Rc &= A \times B \\ &= \int_{X \times Y} \mu_A(x) \wedge \mu_B(y) / (x, y) \end{aligned}$$

当 x 为 A'，且模糊关系的合成运算采用“极大–极小”运算时，模糊推理的结论计算如下：

$$\begin{aligned} B' &= A' \circ R_c \\ &= \int_Y \bigvee_{x \in X} (\mu_{A'}(x) \wedge (\mu_A(x) \wedge \mu_B(y))) / y \end{aligned}$$

- **Larsen 模糊推理算法**

Larsen 模糊推理算法采用乘积运算作为模糊蕴含的规则，用来定义相应的模糊关系。设规则

“IF x 为 A THEN y 为 B”

表达的模糊关系为 R_p，则 R_p 的计算如下：

$$\begin{aligned} R_p &= A \times B \\ &= \int_{X \times Y} \mu_A(x) \mu_B(y) / (x, y) \end{aligned}$$

当 x 为 A'，且模糊关系的合成运算采用“极大–极小”运算时，模糊推理的结论计算如下：

$$B' = A' \circ R_c = \int_Y \bigvee_{x \in X} (\mu_{A'}(x) \wedge (\mu_A(x) \cdot \mu_B(y))) / y$$

- Takagi－Sugeno 型模糊推理

与其他类型的模糊推理方法不同，Takagi–Sugeno 型模糊推理将去模糊化也结合到模糊推理中，其输出为精确量。这是由 Takagi–Sugeno 型模糊规则的形式所决定的，在 Sugeno 型模糊规则的后件部分将输出量表示为输入量的线性组合，零阶 Sugeno 型模糊规则具有如下形式：

IF x 为 A 并且 y 为 B THEN $z=k$

其中，k 为常数。而一阶 Sugeno 型模糊规则的形式如下：

IF x 为 A 并且 y 为 B THEN $z=p*x+q*y+r$

上式中，p、q、r 均为常数。对于一个由 n 条规则组成的 Sugeno 型模糊推理系统，设每条规则具有下面的形式：

Ri: IF x 为 A_i 并且 y 为 B_i THEN $z=z_i$ $(i=1,2,...n)$

则系统总的输出用下式计算：

$$y = \frac{\sum_{i=1}^{n} \mu_{Ai}(x)\mu_{Bi}(y)z_i}{\sum_{i=1}^{n} \mu_{A_i}(x)\mu_{B_i}(y)}$$

2 有关函数介绍

- **evalfis**

功能： 执行模糊推理计算。

格式： output = evalfis(input,fismat)

说明： 该函数计算以 input 为输入模糊向量的模糊推理系统的输出模糊向量 output。fismat 为模糊推理系统的矩阵名称。Evalfis 有两种文件格式，即 M-文件和 MEX-文件，考虑到运算的速度，通常调用 MEX-文件执行模糊推理计算。输入向量是 $M \times N$ 矩阵，其中 N 是输入变量个数；输出向量是 $M \times L$ 矩阵，其中 L 是输出语言变量个数。

举例：

```
fismat = readfis('tipper');
out = evalfis([2 1; 4 9],fismat)
输出结果:
out =
7.0169 19.681
```

- defuzz

功能：执行输出去模糊化。

格式：out = defuzz(x,mf,type)

说明：参数x是变量的论域范围，mf为待去模糊化的模糊集合，type是去模糊化的方法。去模糊化的方法包括5种，即centroid（面积中心法）、bisector（面积平分法）、mom（平均最大隶属度方法）、som（最大隶属度中的取最小值方法）、lom（最大隶属度中的取最大值方法）。

(1) centroid（面积中心法）

面积中心方法又称为重心法，即计算隶属度函数曲线包围区域的重心。对于连续论域的情形，设$\underset{\sim}{U}$是某一变量u在论域U上的模糊集合，则去模糊化的结果为

$$u_c = \frac{\int_U \underset{\sim}{U}(u)u\,du}{\int_U \underset{\sim}{U}(u)\,du}$$

(2) bisector（面积平分法）

面积平分方法即计算将隶属度函数曲线包围面积平分为两部分的某一点，并取该点为去模糊化的结果。

(3) mom（平均最大隶属度方法）

平均最大隶属度方法取模糊集合中具有最大隶属度点的平均值作为去模糊化的结果。

(4) som（最大隶属度取最小值方法）

该方法取模糊集合中具有最大隶属度的所有点中的最小的一个作为去模糊化的结果。

(5) lom（最大隶属度取最大值方法）

该方法取模糊集合中具有最大隶属度的所有点中的最大的一个作为去模糊化的结果。

举例：
```
x = -10:0.1:10;
mf = trapmf(x,[-10 -8 -4 7]);
xx = defuzz(x,mf,'centroid');
```
输出为：
```
xx=-3.2857
```

- gensurf

功能：生成模糊推理系统的输出曲面并显示。

格式：gensurf(fis)

gensurf(fis,inputs,output)

gensurf(fis,inputs,output,grids,refinput)

说明：参数 fis 为模糊推理系统对应的矩阵；inputs 为模糊推理系统的一个或两个输入语言变量；output 为模糊推理系统的输出语言变量；参数 grids 用于指定 X 和 Y 坐标方向的网格数目；当系统输入变量多于两个时，参数 refinput 用于

指定保持不变的输入变量。当仅有一个参数 fis 时，该函数生成由模糊推理系统的前两个输入和第一个输出构成的三维曲面。

举例:

```
a = readfis('tipper');
gensurf(a)
```

输出曲面如图5.3.13所示。

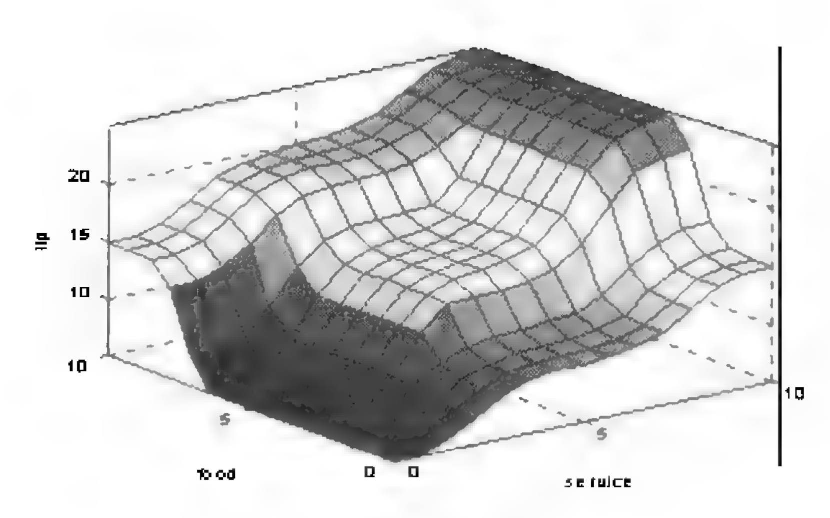


图5.3.13 模糊推理系统输出特性曲面图

5.4 模糊逻辑工具箱的图形界面工具

前面介绍了模糊逻辑工具箱中有关构造模糊推理系统的函数，这些函数都是直接在 Matlab 命令行窗口执行并显示结果的。为了进一步方便用户，模糊逻辑工具箱提供了一套图形界面的模糊推理系统构造函数。这类函数共有五个，如表 5.4.1 所示。

表 5.4.1 模糊逻辑工具箱的图形界面函数

函数名称	功能
fuzzy	基本模糊推理系统编辑器
Mfedit	隶属度函数编辑器
ruleedit	模糊推理规则编辑器
ruleview	模糊推理规则观察器
surfview	模糊推理输出特性曲面观察器

5.4.1 基本模糊推理系统编辑器（Fuzzy）

基本模糊推理系统编辑器提供了利用图形界面（GUI）对模糊系统的高层属性的编辑、修改功能，这些属性包括输入、输出语言变量的个数和去模糊化方法等。用户在基本模糊编辑器中可以通过菜单选择激活其他几个图形界面编辑器，如模糊规则编辑器（ruleedit）、隶

属度函数编辑器(mfedit)等。在Matlab命令窗口执行fuzzy命令即可激活基本模糊推理系统编辑器，其图形界面如图5.4.1 所示。

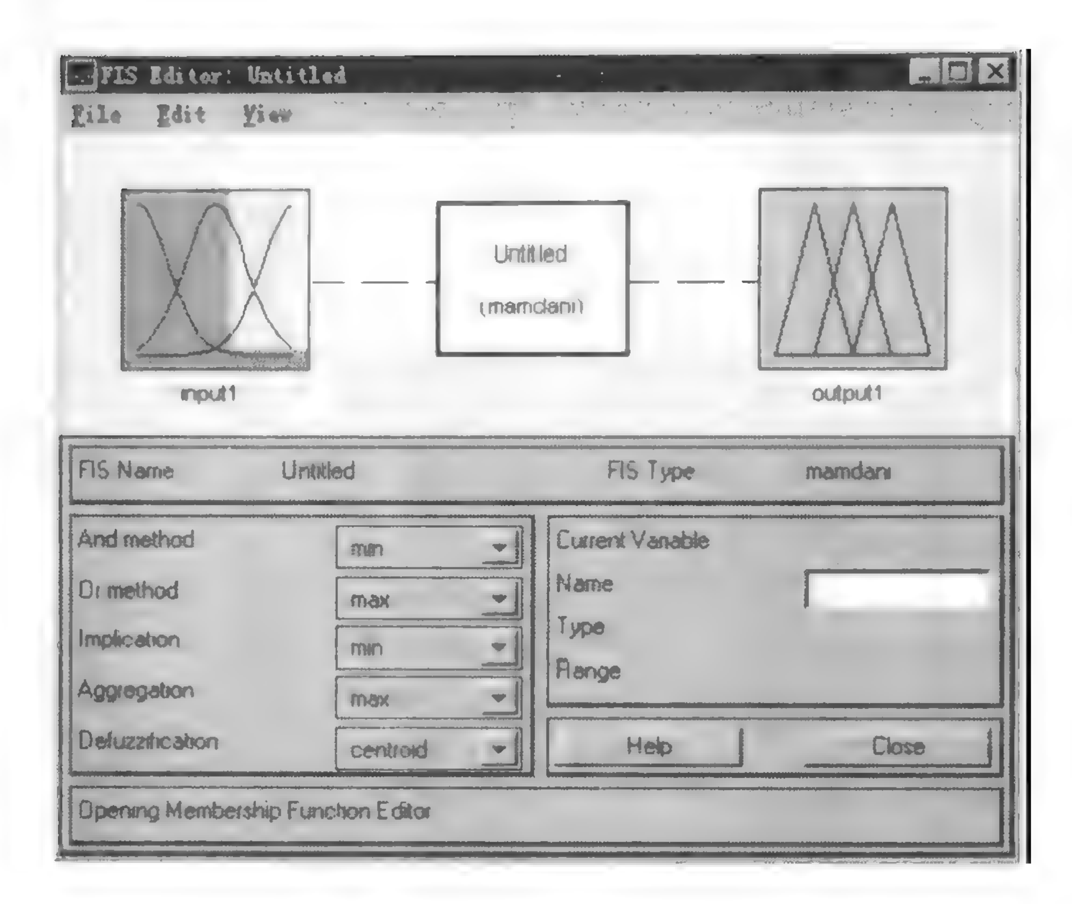


图5.4.1　基本模糊推理系统编辑器图形界面

从图5.4.1中可以看到，在窗口上半部以图形框的形式列出了模糊推理系统的基本组成部分，即输入模糊变量、模糊规则和输出模糊变量。通过鼠标双点上述图形框，能够激活隶属度函数编辑器和模糊规则编辑器等相应的编辑窗口。在窗口的下半部分的左侧列出了模糊推理系统的名称、类型和一些基本属性，包括“与”运算方法、“或”运算方法、蕴含运算、模糊规则的综合运算以及去模糊化方法等。用户只需用鼠标即可设定相应的属性。在图5.4.1中，模糊推理系统的基本属性设定为：“与”运算采用极小运算，“或”运算采用极大运算，模糊蕴含采用极小运算，模糊规则综合采用极大运算，去模糊化采用重心法。窗口下半部分的右侧，列出了当前选定的模糊语言变量的名称及其论域范围。

在fuzzy的菜单部分主要提供了如下功能。

1　文件（File）菜单

文件菜单的主要功能包括：

• New Mamdani FIS——新建Mamdani型模糊推理系统；

• New Sugeno FIS——新建Sugeno型模糊推理系统；

• Open FIS From Disk——从磁盘打开一个模糊推理系统文件；

• Save to disk——将当前的模糊推理系统保存到磁盘文件中；

- Save to disk as——将当前的模糊推理系统另存为一个文件；
- Open FIS from Workspace——从工作空间加载一个模糊推理系统；
- Save to Workspace——保存到工作空间；
- Save to Workspace as——另存到工作空间的某一模糊推理系统矩阵中；
- Print——打印模糊推理系统的信息；
- Close window——关闭窗口。

2 编辑菜单

编辑菜单的功能包括：

- Add input——添加输入语言变量；
- Add output——添加输出语言变量；
- Remove variable——删除语言变量。

3 视图(View)菜单

视图菜单的功能包括：

- Edit FIS Properties——修改模糊推理系统的特性；
- Edit membership functions——打开隶属度函数编辑器；
- Edit Rules——打开模糊规则编辑器；
- View Rules——打开模糊规则浏览器；
- View Surface——打开模糊系统输入输出特性浏览器。

5.4.2 隶属度函数编辑器（Mfedit）

在命令窗口键入mfedit或在基本模糊推理系统编辑器中选择编辑隶属度函数菜单，都可激活隶属度函数编辑器。在该编辑器中，提供了对输入输出语言变量各语言值的隶属度函数类型、参数进行编辑、修改的图形界面工具，其界面如图5.4.2所示。

在该图形界面中，窗口上半部分为隶属度函数的图形显示，下半部分为隶属度函数的参数设定界面，包括语言变量的名称、论域和隶属度函数的名称、类型和参数。

在菜单部分，文件菜单和视图菜单的功能与模糊推理系统编辑器的文件功能类似。编辑菜单的功能包括添加隶属度函数、添加定制的隶属度函数以及删除隶属度函数等。

5.4.3 模糊规则编辑器（Ruleedit）

在Matlab命令窗口键入Ruleedit或在基本模糊推理系统编辑器中选择编辑模糊规则菜单，均可激活模糊规则编辑器。在模糊规则编辑器中，提供了添加、修改和删除模糊规则的图形界面，如图5.4.3所示。

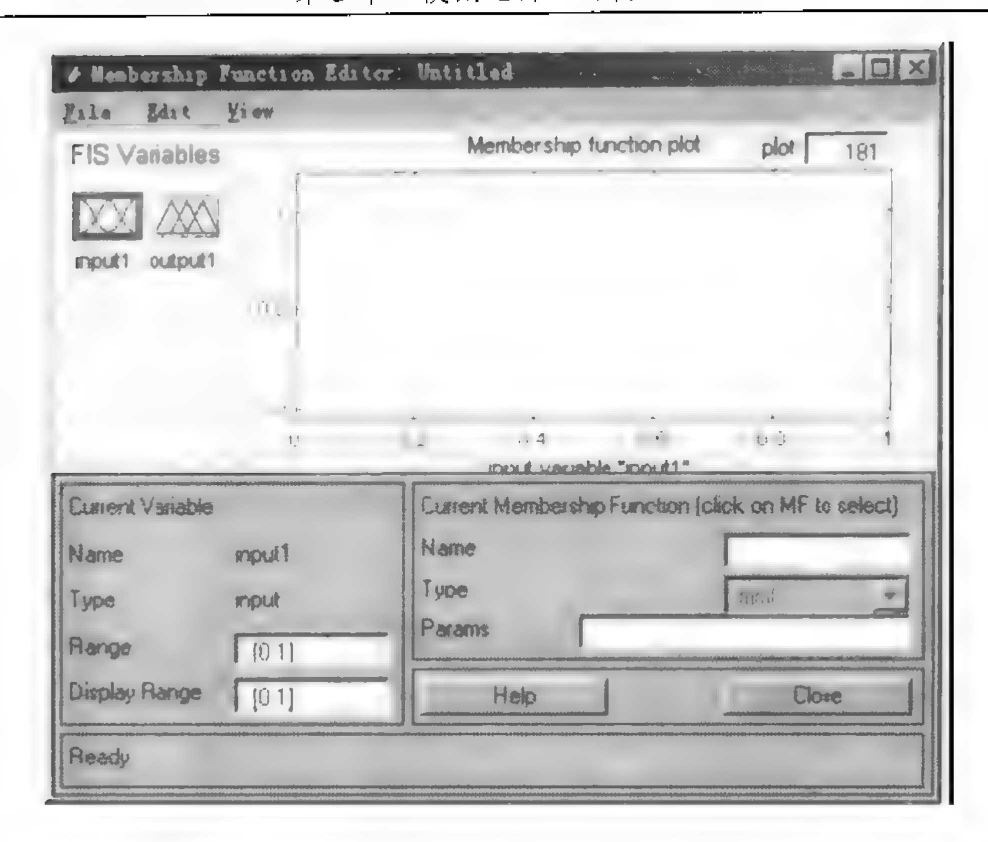


图5.4.2 隶属度函数编辑器



图5.4.3 模糊规则编辑器

在模糊规则编辑器中提供了一个文本编辑窗口，用于规则的输入和修改。模糊规则的形式可有三种，即语言型（Verbose）、符号型（Simbolic）以及索引型(Indexed)。在窗口的下部有一个下拉列表框，供用户选择某一规则类型。

模糊规则编辑器的菜单功能与前两种编辑器基本类似，在其视图菜单中能够激活其他的编辑器或窗口。

5.4.4 模糊规则浏览器（Ruleview）

在Matlab命令窗口键入ruleview或在上述三种编辑器中选择相应菜单，都可激活模糊规则浏览器。在模糊规则浏览器中，以图形形式描述了模糊推理系统的推理过程，其界面如图5.4.4所示。

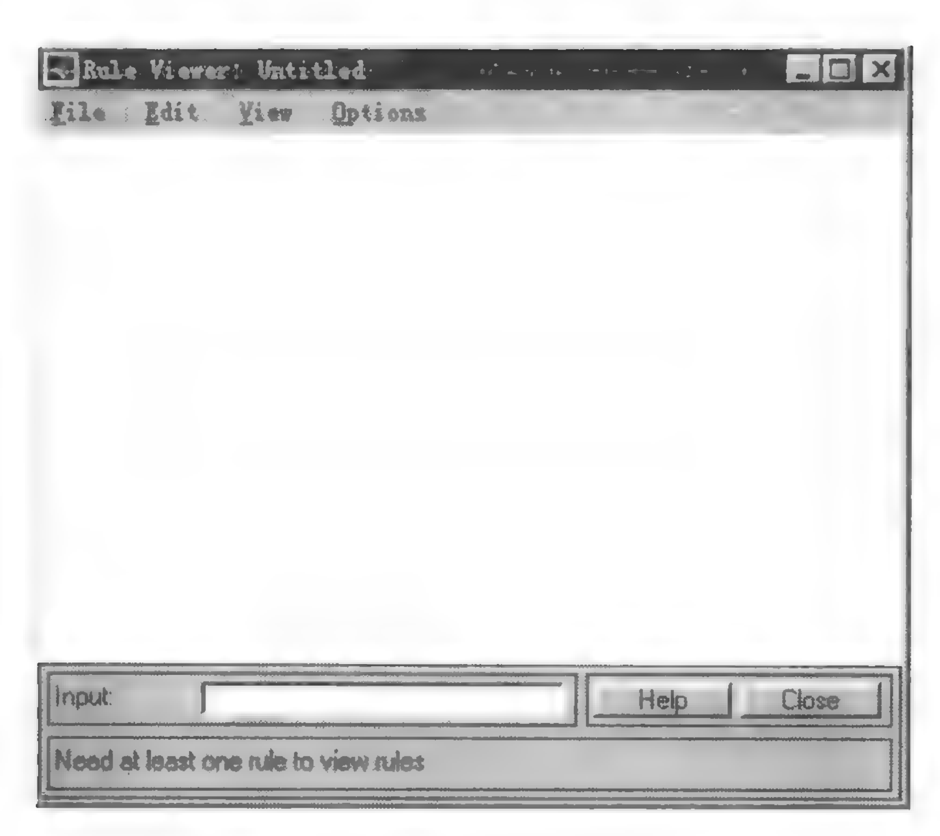


图5.4.4 模糊规则浏览器

5.4.5 模糊推理输入输出曲面视图(Surfview)

在Matlab命令窗口键入Surfview命令或在各个编辑器窗口选择相应菜单，即可打开模糊推理的输入输出曲面视图窗口。该窗口以图形的形式显示模糊推理系统的输入输出特性曲面。

5.5 Matlab 模糊逻辑工具箱的高级应用

在 Matlab 模糊逻辑工具箱中，提供了有关模糊逻辑推理的高级应用，包括自适应神经模糊推理系统、模糊聚类、给定数据的模糊建模。相关的函数如表 5.5.1 所示。

表 5.5.1　模糊推理系统的高级应用

函数名称	功能
anfis	利用 Sugeno 型模糊推理的神经模糊推理系统建模
fcm	采用模糊 C 均值方法的模糊聚类
genfis1	基于网格分割方法的模糊推理系统建模
genfis2	基于减聚类方法的模糊推理系统建模
anfisedit	神经模糊推理系统建模的图形界面工具
subclust	数据的模糊减聚类

5.5.1 基于 Sugeno 型模糊推理的神经模糊系统建模

在前面对模糊推理的讨论中，分别介绍了 Mamdani 型模糊推理和 Takagi–Sugeno 型模糊推理。Matlab 模糊逻辑工具箱同时提供对这两种模糊推理方法的支持。其中对 Sugeno 型模糊推理仅支持一阶规则，即规则的输出为输入变量的线性组合。

Mamdani 型模糊推理和 Takagi–Sugeno 型模糊推理各有优缺点。对 Mamdani 型模糊推理，由于其规则的形式符合人们思维和语言表达的习惯，因而能够方便地表达人类的知识，但存在计算复杂、不利于数学分析的缺点；而 Takagi–Sugeno 型模糊推理则具有计算简单，利于数学分析的优点，且易于和 PID 控制方法以及优化、自适应方法结合，从而实现具有优化与自适应能力的控制器或模糊建模工具。

根据 Sugeno 型模糊推理的特点，有关学者将其与神经网络结合，用于构造具有自适应学习能力的神经模糊系统。模糊逻辑与神经网络的结合是近年来计算智能学科的一个重要研究方向。两者结合形成的模糊神经网络同时具有模糊逻辑易于表达人类知识和神经网络的分布式信息存储以及学习能力的优点，对于复杂系统的建模和控制提供了有效的工具。

在 Matlab 模糊逻辑工具箱中，提供了对基于 Sugeno 型模糊推理的神经模糊系统的支持，该模糊推理系统利用反向传播算法和最小二乘方算法来完成对输入–输出数据对的建模。相应的函数为 anfis。anfis 支持采用输出加权平均的一阶 Sugeno 型模糊推理，该函数的说明如下。

1 anfis

功能：利用 Sugeno 型神经模糊系统的建模。

格式：[fismat1,error1,stepsize] = anfis(trnData)

[fismat1,error1,stepsize] = anfis(trnData,fismat)

```
[fismat1,error1,stepsize]=
    anfis(trnData,fismat,trnOpt,dispOpt)
[fismat1,error1,stepsize,fismat2,error2] =
    anfis(trnData,trnOpt,dispOpt,chkData)
[fismat1,error1,stepsize,fismat2,error2] =
    anfis(trnData,fismat ,trnOpt,dispOpt,chkData)
```

说明：该函数的输出为一个三维或五维向量。当未指定检验数据时，输出向量为三维。其中，参数fismat1为学习完成后得到的对应最小均方根误差的模糊推理系统矩阵；error1为训练的均方根误差向量；stepsize为训练步长向量。当指定检验数据后，输出向量为五维参数向量，参数fismat2为对检验数据具有最小均方根误差的模糊推理系统，error2为检验数据对应的最小均方根误差。

在函数的输入参数向量中，trnData为训练学习的输入输出数据矩阵。该矩阵的每一行对应一组输入输出数据，其中最后一列为输出数据；fismat是指定初始的模糊推理系统参数（包括隶属度函数类型和参数）的矩阵，该矩阵可以使用函数fuzzy通过模糊推理系统编辑器生成，也可使用函数genfis1由训练数据直接生成。函数genfis1的功能采用网格分割法生成模糊推理系统，其使用方法参见下文的说明。

函数anfis的输入参数中，trnOpt和dispOpt分别指定训练的有关选项和在训练执行过程中Matlab命令窗口的显示选项。参数trnOpt为一个五维向量，其各个分量的定义如下：

- trnOpt(1)——训练的次数，缺省为10；
- trnOpt(2)——期望误差，缺省为0；
- trnOpt(3)——初始步长，缺省为0.01；
- trnOpt(4)——步长递减速率，缺省为0.9；
- trnOpt(5)——步长递增速率，缺省为1.1。

如果trnOpt的任一个分量为NaN(非数值：IEEE的标准缩写)或被省略，则训练采用缺省参数。学习训练的过程在训练参数得到指定值或训练误差得到期望误差时停止。训练过程中的步长调整采用如下的策略：

- 当误差连续四次减小时，则增加步长；
- 当误差变化连续两次出现振荡即一次增加和一次减少交替发生，则减小步长。

TrnOpt的第四和第五个参数分别按照上述策略控制训练步长的调整。

参数dispOpt用于控制训练过程中Matlab命令窗口的显示内容，共有四个参数，分别定义如下：

- dispOpt(1)——显示ANFIS的信息，缺省为1；
- dispOpt(2)——显示误差测量，缺省为1；
- dispOpt(3)——显示训练步长，缺省为1；
- dispOpt(4)——显示最终结果，缺省为1。

当上述某一分量为0时，则不显示相应的内容；如果为1或为NaN或省略，则显

示相应内容。

函数anfis的另一个输入参数为chkData，该参数为一个与训练数据矩阵有相同列数的矩阵，用于提供检验数据。当提供检验数据时，anfis返回对检验数据具有最小均方根误差的模糊推理系统fismat2。

举例：

```
x = (0:0.1:10);
y = sin(2*x)./exp(x/5);
trnData = [x' y'];
numMFs = 5;
mfType = 'gbellmf';
epoch_n = 20;
in_fismat = genfis1(trnData,numMFs,mfType);
out_fismat = anfis(trnData,in_fismat,20);
plot(x,y,'-',x,evalfis(x',out_fismat),':');
legend('Training Data','ANFIS Output');
```

Matlab的输出曲线如图5.5.1所示。

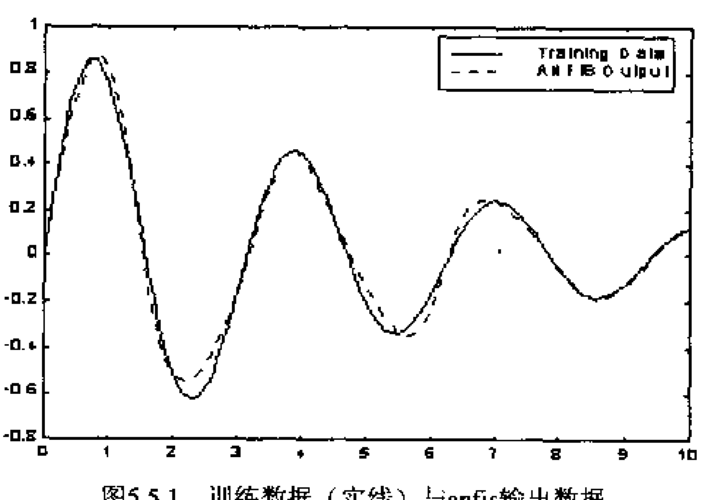


图5.5.1 训练数据（实线）与anfis输出数据

2 anfis的详细应用实例

为进一步说明基于Sugeno型模糊推理的神经模糊系统建模及anfis的用法，这里给出一个更具体的应用例子。

在这个例子中，不但提供了训练数据，而且提供了检验数据，两种数据在输入空间均匀采样，图5.5.2显示了训练数据和检验数据的分布。

```
% 数据点个数为51
numPts = 51;
x1 = linspace(0,1,numPts)';
y = 0.6*sin(pi*x1) + 0.3*sin(3*pi*x1) + 0.1*sin(5*pi*x1);
data = [x1 y]; % 整个数据集
```

```
trnData = data(1:2:numPts,:); % 训练数据集
chkData = data(2:2:numPts,:); % 检验数据集
%绘制训练数据和检验数据的分布曲线
 plot(trnData(:,1),trnData(:,2),'o', ...
 chkData(:,1),chkData(:,2),'x')
%建立用于模糊建模的Sugeno型模糊推理系统
%采用genfis1函数直接由训练数据生成模糊推理系统
 numMFs = 5; % 隶属度函数个数
 mfType = 'gbellmf'; % 隶属度函数类型
 fismat = genfis1(trnData,numMFs,mfType);
 %绘制模糊推理系统的初始隶属度函数
 [x,mf]=plotmf(fismat,'input',1);
 plot(x,mf)
 title('Initial Membership Functions')
```

图 5.5.3 显示由 genfis1 根据训练数据生成的模糊推理系统隶属度函数曲线。从曲线可以看出，函数 genfis1 按照均匀覆盖输入空间的原则构造了初始隶属度函数。

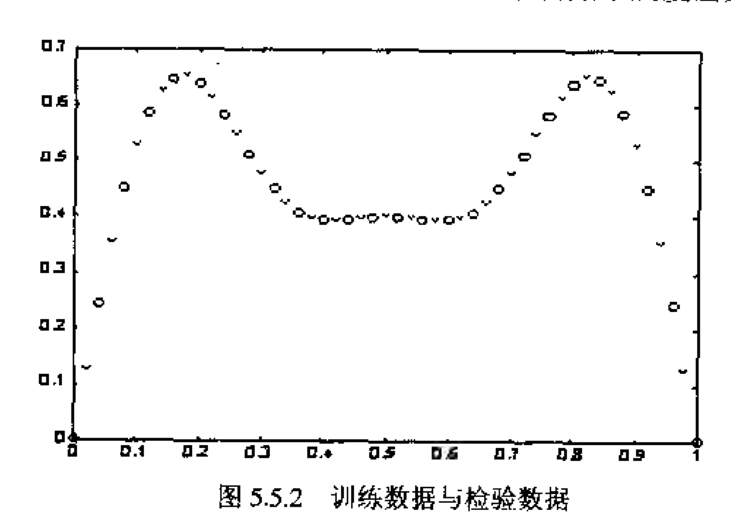


图 5.5.2 训练数据与检验数据

图 5.5.3 anfis 的初始隶属度函数

下面使用anfis函数进行给定数据的神经模糊建模。

```
numEpochs = 40;      %训练次数为40
[fismat1,trnErr,ss,fismat2,chkErr] =
       anfis(trnData,fismat,numEpochs,NaN,chkData);
%计算训练后神经模糊系统的输出与训练数据的均方根误差
trnOut = evalfis(trnData(:,1),fismat1);
trnRMSE = norm(trnOut-trnData(:,2))/sqrt(length(trnOut));
%绘制训练过程中均方根误差的变化情况，如图5.5.4所示。
```

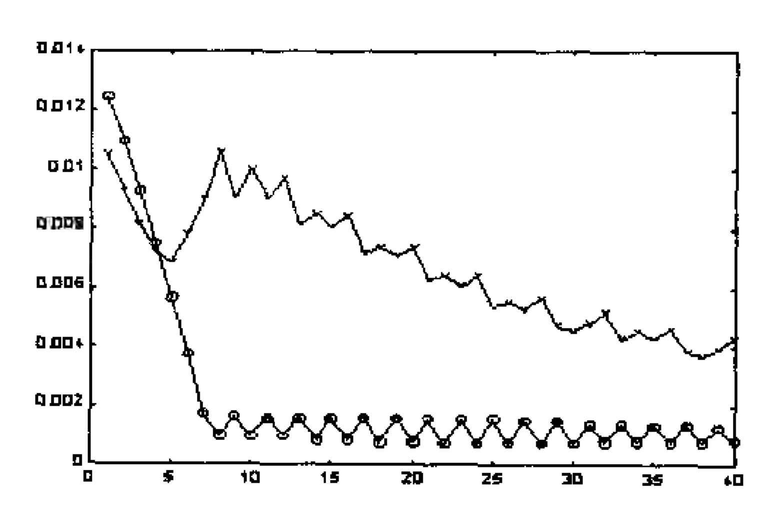


图 5.5.4　训练过程中均方根误差的变化曲线

```
epoch = 1:numEpochs;
plot(epoch,trnErr,'o',epoch,chkErr,'x')
hold on; plot(epoch,[trnErr chkErr]); hold off
%绘制训练过程中步长的变化情况，如图5.5.5所示。
```

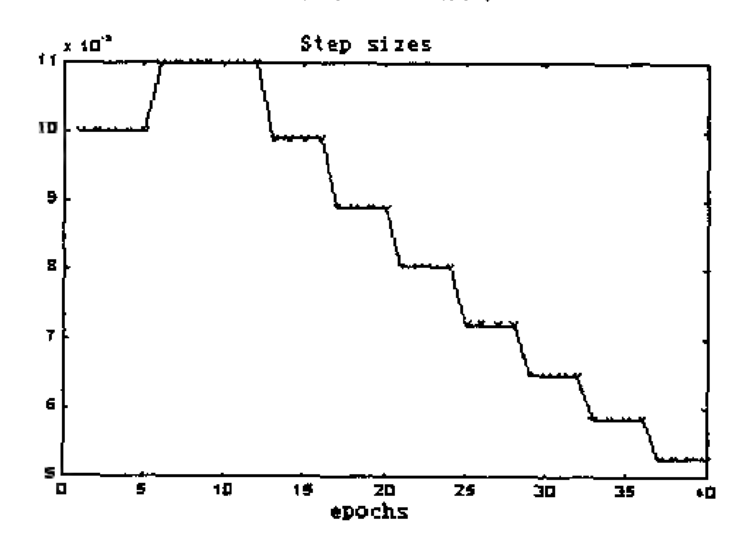


图 5.5.5　训练步长变化曲线

```
plot(epoch,ss,'-',epoch,ss,'x')
xlabel('epochs ), ylabel('ss'), title('Step Sizes')
```

%下面绘制训练后模糊推理系统的隶属度函数曲线，如图 5.5.6 所示。

```
[x,mf]=plotmf(fismat1,'input',1);
plot(x,mf)
title('Final Membership Functions');
```

从图 5.5.6 中可以看出，经过学习后的模糊推理系统提取了训练数据的局部特征。下面绘制神经模糊推理系统的输出曲线，如图 5.5.7 所示。

```
anfis_y=evalfis(x,fismat1);
plot(x1,y,'-',x,anfis_y,'x');
```

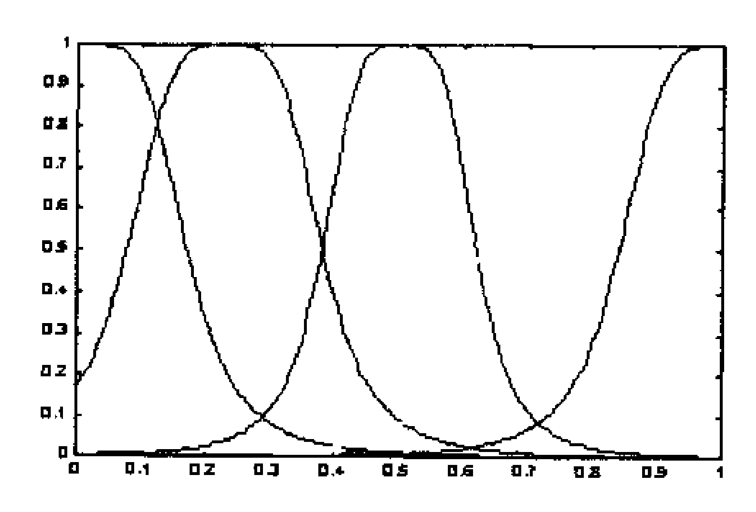


图 5.5.6　训练后模糊推理系统的隶属度函数

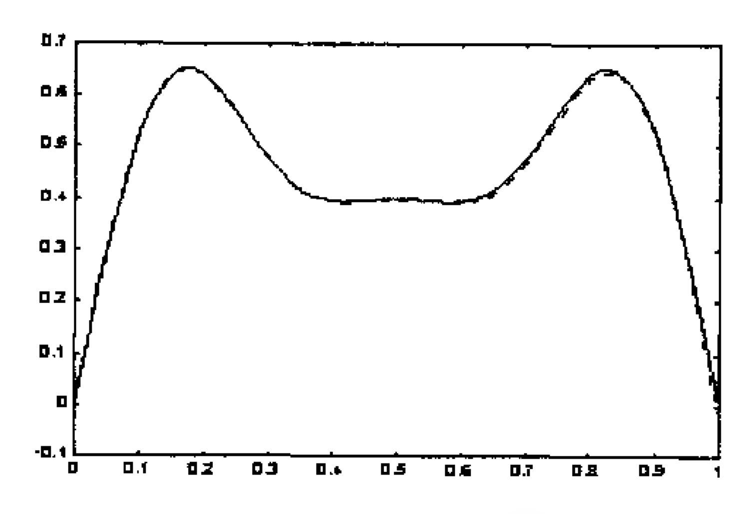


图 5.5.7　神经模糊推理系统的输入输出特性与给定数据的比较

3　采用网格分割方式生成模糊推理系统

在给定输入输出数据后，要利用anfis函数进行神经模糊系统建模，必须提供一个以输入输出数据为基础的初始模糊推理系统。函数genfis1采用网格分割的方式根据给定数据集生成一个模糊推理系统，因而可以与函数anfis配合使用。由genfis1生成的模糊推理系统的输入和输出隶属度函数曲线都在保证覆盖整个输入输出空间的基础上进行均匀分割，其输入输出隶

属度函数的类型和数目可以在使用时指定，也可以采用缺省值。函数genfis1的使用格式说明如下：

- genfis1

功能：采用网格分割方式生成模糊推理系统。

格式：fismat = genfis1(data)

fismat = genfis1(data,numMFs,mfType)

说明：输入参数中，data为给定的输入输出数据集合，numMFs为一个整数向量，用于指定输入、输出语言变量的隶属度函数个数；参数mfType用于指定隶属度函数的类型。fismat为生成的模糊推理系统矩阵。当仅使用一个输入参数而不指定隶属度函数个数和类型时，将使用缺省值，即隶属度函数个数为2，类型为钟型曲线。

举例：

```
data = [rand(10,1) 10*rand(10,1)-5 rand(10,1)];
numMFs = [3 7];
mfType = str2mat('pimf','trimf');
fismat = genfis1(data,numMFs,mfType);
[x,mf] = plotmf(fismat,'input',1);
subplot(2,1,1), plot(x,mf);
xlabel('input 1 (pimf)');
[x,mf] = plotmf(fismat,'input',2);
subplot(2,1,2), plot(x,mf);
xlabel('input 2 (trimf)');
```

输出的隶属度函数曲线如图5.5.8所示。

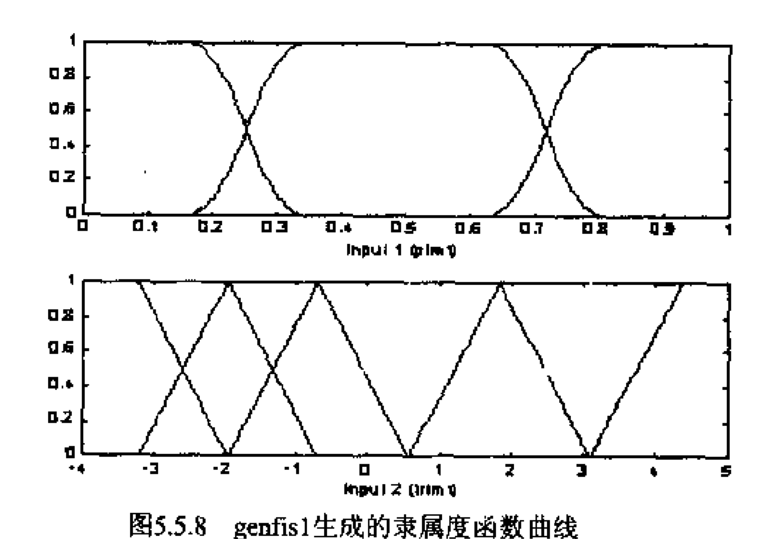


图5.5.8　genfis1生成的隶属度函数曲线

4　神经模糊推理系统的图形界面工具

为进一步方便用户的使用，在模糊逻辑工具箱中提供了建立神经模糊推理系统的图形界

面工具，该工具以交互式图形界面的形式集成了建立、训练和测试神经模糊推理系统等各种功能。要启动该工具，只需在Matlab命令窗口键入‘anfisedit’即可得到如图5.5.9所示的图形窗口界面。

该图形界面包括的功能主要有：加载数据（Load Data）、生成模糊推理系统（Generate FIS）、训练神经模糊推理系统(Train FIS)以及测试神经模糊推理系统。

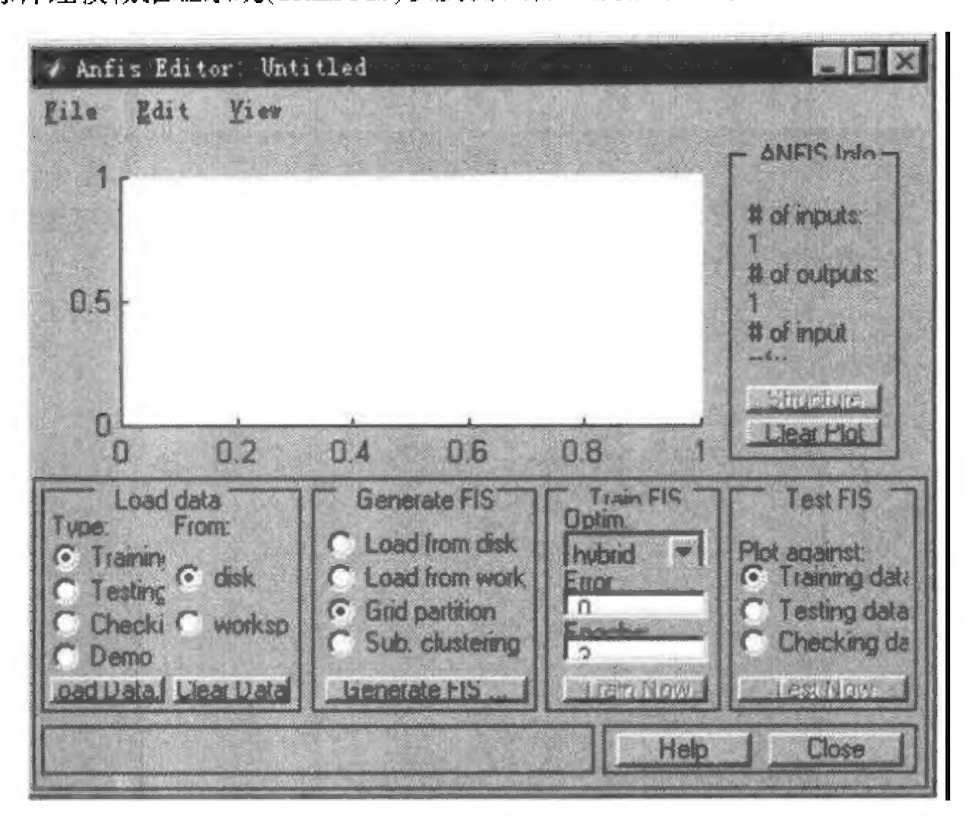


图5.5.9　anfis图形界面

下面以一个实例来说明该图形界面的使用方法。

- **加载数据**

通过界面上的检查框可以选择加载数据的类型，如训练数据、测试数据和演示数据等。为方便说明，这里选择加载演示数据。加载了演示数据的图形窗口显示如图5.5.10所示。在图5.5.10的上部显示了数据的变化情况。

- **生成模糊推理系统**

在生成模糊推理系统时，也可通过检查框选择模糊推理系统的生成方法，如网格分割法、减法聚类方法等。当用鼠标单击生成模糊推理系统的功能按钮时，系统会弹出一个对话框，要求指定模糊推理系统的有关信息，如图5.5.11所示。

在图5.5.11的对话框中，要求输入的信息包括输入语言变量隶属度函数的数目、类型和输出隶属度函数的类型等。

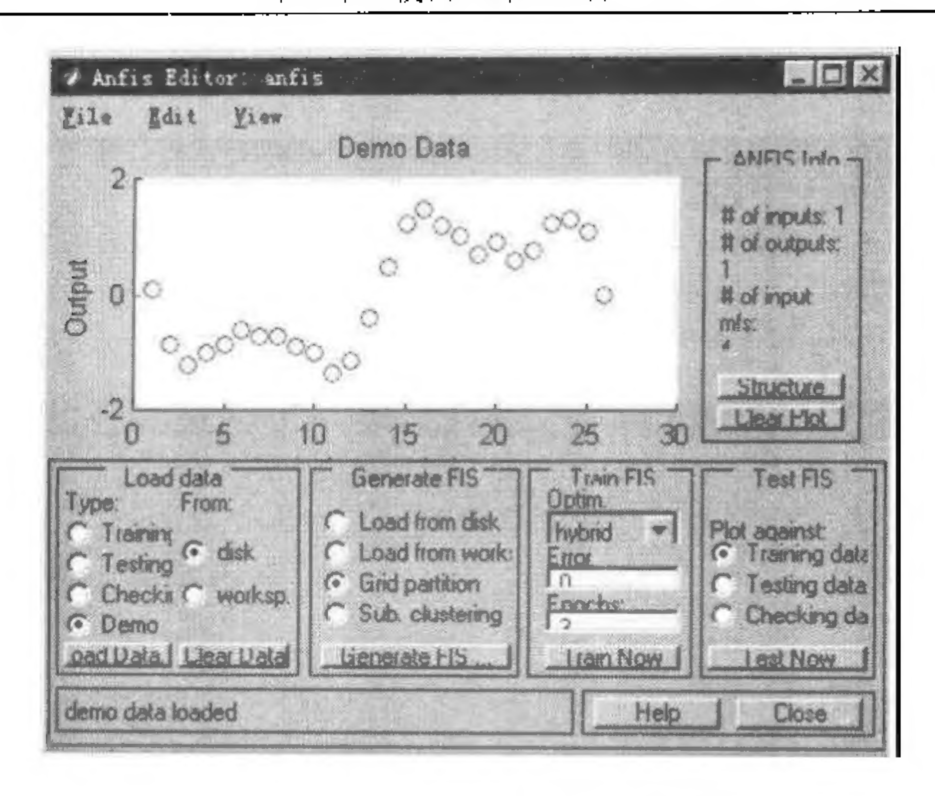


图5.5.10　加载了演示数据的anfis图形界面

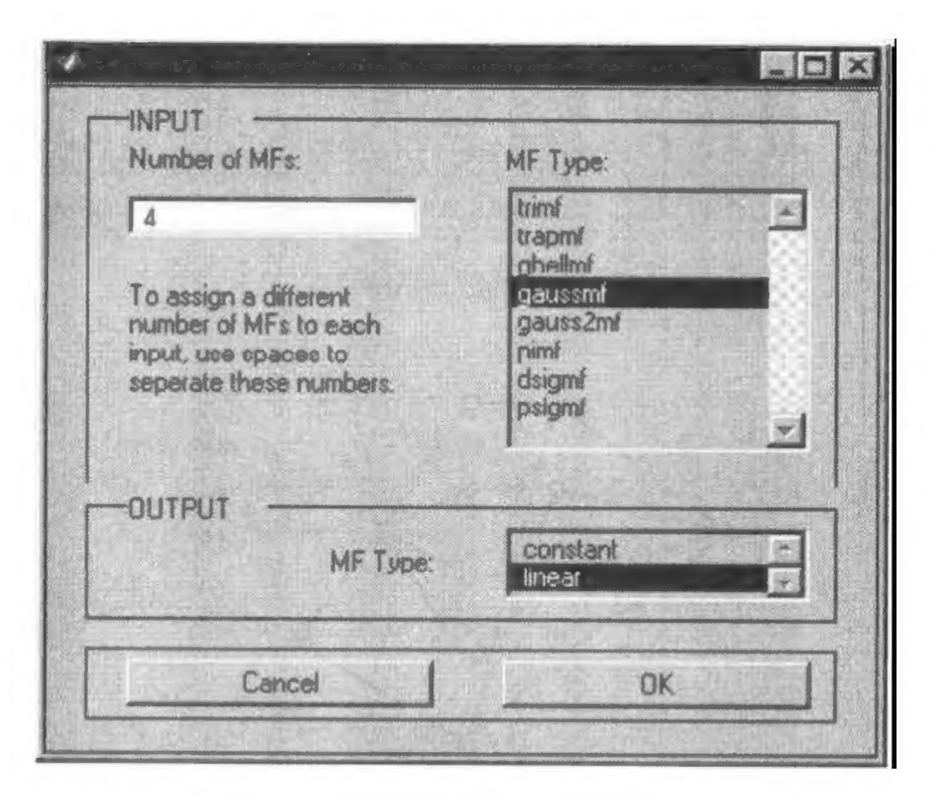


图5.5.11　生成模糊推理系统的对话框

- **训练神经模糊推理系统**

在进行神经模糊推理系统的训练前，可以指定优化的方法以及有关的优化控制参数。对于前面加载的演示数据，在进行anfis的训练后，窗口的上部显示了优化过程中误差的变化情

况，如图5.5.12所示。

- **测试anfis**

在anfis的图形界面中可以方便地查看训练得到的anfis的网络结构。对应上述演示数据的神经网络结构如图5.5.13所示。

在完成对anfis的训练后，可以进一步对其进行测试，测试数据可以被指定为训练数据或另外提供的训练数据。测试完成后将在图形界面的上部显示测试的结果，即anfis输出数据与测试数据的比较，如图5.5.14所示。

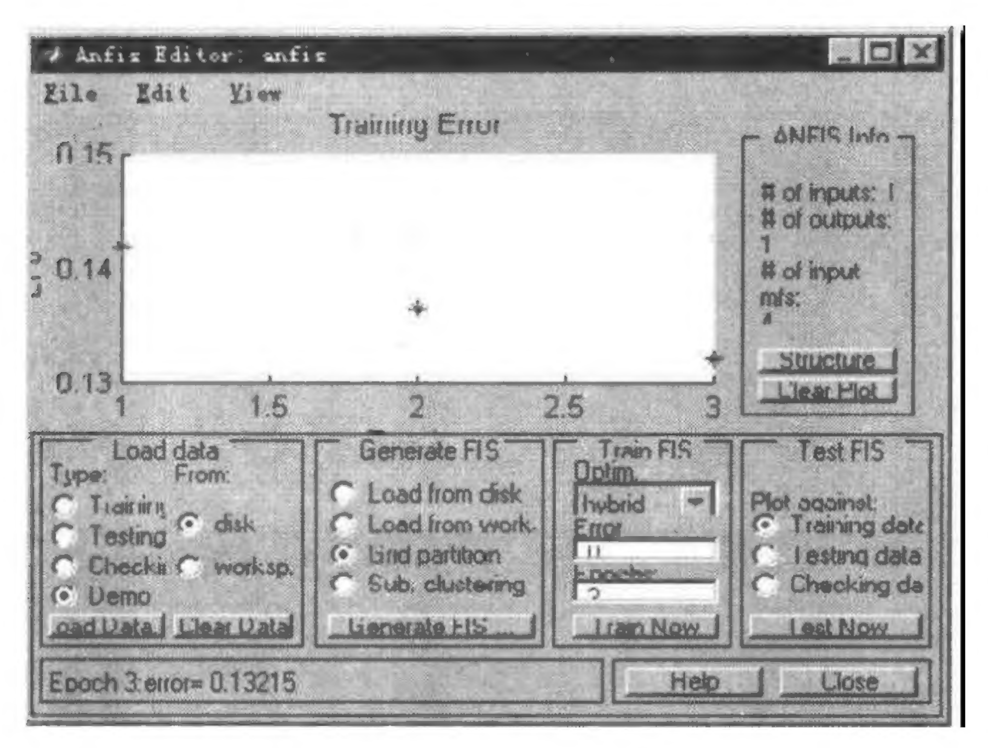


图5.5.12　训练完成后的ANFIS图形界面

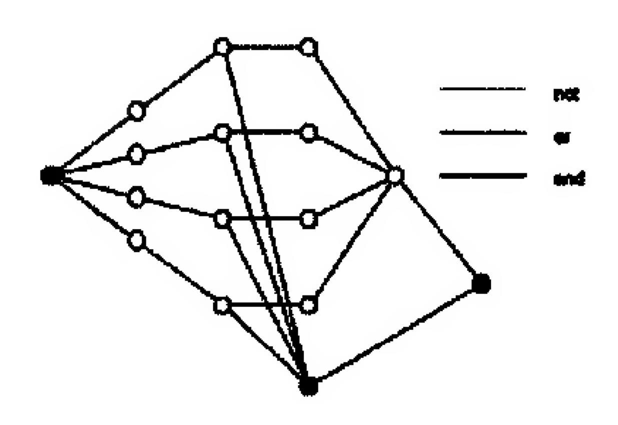


图5.5.13　模糊神经网络的结构

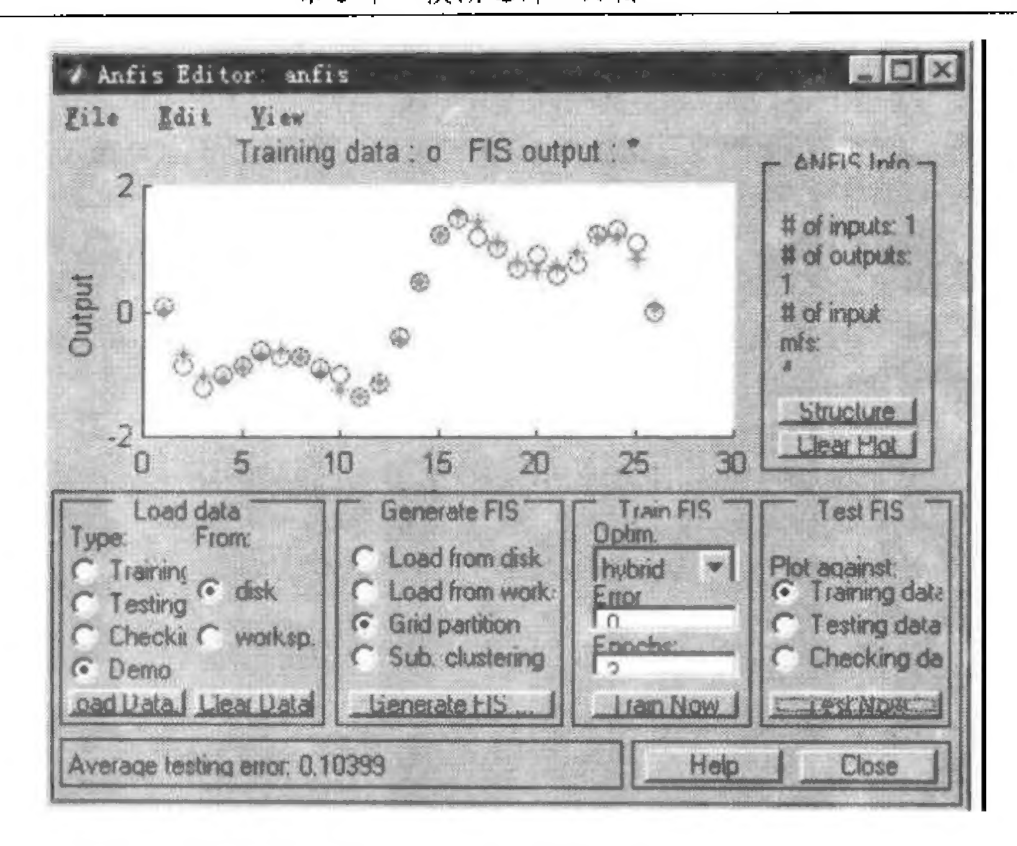


图5.5.14　ANFIS的测试结果

5.5.2　模糊聚类

对给定数据的聚类分析是许多分类和系统建模问题的基础。聚类的目的是从大量的数据中抽取固有的特征，从而获得系统行为的简洁表示。常见的聚类方法有均值聚类、分层聚类和模糊聚类方法等。

在Matlab模糊逻辑工具箱中提供了对两种聚类方法的支持，一种是模糊C-均值聚类方法，另一种是减聚类方法。

1　模糊 C-均值聚类

在模糊C-均值聚类方法中，每一个数据点按照一定的模糊隶属度隶属于某一聚类中心。这一聚类技术作为对传统聚类技术的改进，由Jim Bezdek于1981年提出。该方法首先随机选取若干聚类中心，所有数据点都被赋予对聚类中心一定的模糊隶属度，然后通过迭代方法不断修正聚类中心，迭代过程中以极小化所有数据点到各个聚类中心的距离与隶属度值的加权和为优化目标。

模糊C-均值聚类的输出不是一个模糊推理系统，而是聚类中心的列表以及每个数据点对各个聚类中心的隶属度值。该输出能够被进一步用来建立模糊推理系统。对应模糊C-均值聚类方法的函数为fcm,下面对该函数进行详细说明。

- fcm

功能：模糊C-均值聚类。